TUGAS AKHIR – SM 141501

CONVOLUTIONAL NEURAL NETWORKS UNTUK PENGENALAN WAJAH SECARA REAL-TIME

MUHAMMAD ZUFAR
NRP 1212 100 076

Dosen Pembimbing
Dr. Budi Setiyono, S.Si, MT

JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – SM 141501

CONVOLUTIONAL NEURAL NETWORKS FOR REAL-TIME FACE RECOGNITION

MUHAMMAD ZUFAR
NRP 1212 100 076

Supervisor
Dr. Budi Setiyono, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Science
Sepuluh Nopember Institute of Technology
Surabaya 2016

LEMBAR PENGESAHAN
CONVOLUTIONAL NEURAL NETWORKS UNTUK PENGENALAN WAJAH SECARA REAL-TIME

CONVOLUTIONAL NEURAL NETWORKS FOR REAL-TME FACE RECOGNITION

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Sains
Pada bidang studi Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
MUHAMMAD ZUFAR
NRP. 1212 100 076

Menyetujui,
Dosen Pembimbing,

Dr. Budi Setiyono, S.Si, MT
NIP. 19720207 199702 1 001

Mengetahui,
Ketua Jurusan Matematika,

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, 28 Juli 2016

CONVOLUTIONAL NEURAL NETWORKS UNTUK PENGENALAN WAJAH SECARA REAL-TIME

Nama : **Muhammad Zufar**
NRP : **1212 100 076**
Jurusan : **Matematika**
Dosen Pembimbing : **Dr. Budi Setiyono, S.Si, MT**

ABSTRAK

Identifikasi identitas individu melalui pengenalan wajah secara otomatis merupakan suatu persoalan besar yang menarik dan banyak sekali berbagai macam pendekatan untuk menyelesaikan persoalan ini. Apalagi di dalam skenario kehidupan nyata yang tidak terkontrol, wajah akan terlihat dari berbagai sisi dan tidak selalu menghadap ke depan yang membuat permasalahan klasifikasi menjadi lebih sulit diselesaikan. Dalam Tugas Akhir ini digunakan salah satu metode *deep neural networks* yaitu *Convolutional Neural Networks (CNN)* sebagai pengenalan wajah secara *real-time* yang sudah terbukti sangat efisien dalam klasifikasi wajah. Metode diimplementasikan dengan bantuan *library* OpenCV untuk deteksi multi wajah dan perangkat Web Cam M-Tech 5MP. Dalam penyusunan arsitekur model *Convolutional Neural Networks* dilakukan konfigurasi inisialisasi parameter untuk mempercepat proses *training* jaringan. Hasil uji coba dengan munggunakan konstruksi model *Convolutional Neural Networks* sampai kedalaman 7 lapisan dengan input dari hasil ekstraksi *Extended Local Binary Pattern* dengan radius 1 dan *neighbor* 15 menunjukkan kinerja pengenalan wajah meraih rata-rata tingkat akurasi lebih dari 89% dalam $\mp$ 2 frame per detik.

Kata kunci : Pengenalan Wajah, Real-Time, Convolutional Neural Networks

CONVOLUTIONAL NEURAL NETWORKS FOR REAL-TIME FACE RECOGNITION

Name of Student : Muhammad Zufar
NRP : 1212 100 076
Department : Mathematics
Supervisor : Dr. Budi Setiyono, S.Si, MT

ABSTRACT

Identification of individual identity through facial recognition automatically is a huge deal of interest and a lot of the various approaches to resolving the problems. Moreover, in real life scenarios that are not controlled, the face will be visible from all sides and not always facing the front which makes it more difficult classification problems resolved. This final project use one of the methods of deep neural networks that Convolutional Neural Networks (CNN) as face recognition in real-time which has proven very efficient in the face classification. The method is implemented with the help of OpenCV library for multi face detection devices and a 5MP M-Tech Web Cam. In the preparation of architectural models Convolutional Neural Networks do initial configuration parameters to speed up the training process of the network. The trial results with munggunakan model construction Convolutional Neural Networks to a depth of 7 layers with input from the extraction Extended Local Binary Pattern with radius 1 and neighbor 15 shows the performance of face recognition reached average accuracy rate of more than 89% in ∓ 2 frames per second.

Keywords : Face Recognition, Real-Time, Convolutional Neural Networks

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini dijelaskan hal-hal yang melatarbelakangi munculnya permasalahan yang dibahas dalam Tugas Akhir ini. Kemudian permasalahan tersebut disusun kedalam suatu rumusan masalah. Selanjutnya dijabarkan juga batasan masalah untuk mendapatkan tujuan yang diinginkan serta manfaat yang dapat diperoleh. Adapun sistematika penulisan Tugas Akhir ini akan diuraikan di bagian akhir bab ini.

1.1 Latar Belakang

Sebuah sistem autentikasi identitas manusia yang memiliki akurasi tinggi sangat diperlukan saat ini karena meningkatnya jumlah kejahatan dan kerugian melalui penipuan identitas. Sistem berbasis token *(traditional token-based system)* dan berbasis pengetahuan *(knowledge-based system)* memiliki resiko tinggi dalam kasus pencurian atau lupa sandi, oleh karena itu secara luas sistem saat ini menggunakan sistem biometrik seperti kontrol akses, identifikasi kriminal, *autonomous vending*, dan *automated banking* karena keunikan fitur biometrik dan karakteristik yang tidak dapat dipindah tangankan*(non-transferable characteristic)*[1]. Biometrik dibagi menjadi dua kategori yaitu berdasarkan perilaku seseorang seperti ritme mengetik, gaya berjalan atau suara dan berdasarkan fisiologis seperti sidik jari, wajah, iris, atau tanda tangan. Diantara tipe biometrik fisiologis yang disebutkan sebelumnya, pengenalan wajah hingga saat ini masih menjadi daerah penelitian sejak tahun 1960 dengan cakupan ruang yang luas untuk terus dilakukan penyempurnaan.

Pengenalan wajah merupakan permasalahan yang menantang. Salah satu faktornya adalah karena berbagai macam posisi gambar wajah. Kamera dapat menangkap posisi wajah dari depan, samping, atau dari sudut tertentu sehingga menyebabkan beberapa fitur wajah seperti mata atau hidung menjadi tidak terlihat secara penuh. Faktor lain adalah karena ada atau tidak

adanya komponen struktural seperti jenggot, kumis, atau dengan / tanpa kacamata pada gambar wajah. Komponen-komponen struktural memiliki banyak variabilitas termasuk bentuk, warna dan ukuran. Faktor-faktor lain yang dapat mempengaruhi akurasi meliputi pencahayaan, oklusi dan ekspresi wajah. Iluminasi adalah perubahan dari persebaran cahaya karena sifat reflektansi kulit dan kontrol kamera internal yang dapat melemparkan bayangan pada beberapa bagian dari wajah. Oklusi adalah hasil dari suatu objek yang meliputi bagian wajah, seperti syal, sorban, dll. Contoh ekspresi wajah yang tersenyum, tertawa, marah, sedih, terkejut, dan takut[1].

Salah satu teknik yang paling terkenal dalam menangani masalah ini adalah teknik klasifikasi Jaringan Syaraf Tiruan(JST). Jaringan syaraf tiruan adalah adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan saraf manusia. JST merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut dengan kata lain teknik ini memiliki kemampuan untuk belajar dari pengalaman. Jenis model jaringan syaraf tiruan yang memiliki beberapa lapisan disebut sebagai *Multi Layer Perceptron*(MLP) yang menghubungkan secara penuh antar neuronnya memiliki kemampuan klasifikasi yang powerful. Akan tetapi, MLP memiliki beberapa masalah ketika input berupa gambar. Gambar harus dilakukan pre-processing, segmentasi, dan ekstraksi fitur untuk mendapatkan kinerja optimal. Hal ini menyebabkan MLP memiliki banyak parameter bebas atau informasi yang berlebihan dalam arsitektur. Parameter bebas berasal dari pembentukan oleh skema koneksi penuh antara input dan peta fitur dari lapisan yang berhubungan.

Variasi lain dari MLP yang dapat mengatasi permasalahan yang dijelaskan sebelumnya disebut sebagai *Convolutional Neural Networks* (CNN). CNN terinspirasi oleh korteks mamalia visual sel sederhana dan kompleks. Model ini dapat mengurangi

sejumlah parameter bebas dan dapat menangani deformasi gambar input seperti translasi, rotasi dan skala[2].

Berdasarkan penjelasan kelebihan CNN tersebut, dapat diambil kesimpulan bahwa CNN memiliki kemampuan klasifikasi yang diperuntutkan untuk data gambar sehingga pada Tugas Akhir ini model CNN akan digunakan sebagai pengenalan wajah dari berbagai sisi secara real-time.

1.2 Rumusan Masalah

Berdasarkan latar belakang diambil beberapa permasalahan yang digunakan dalam Tugas Akhir ini sebagai berikut:

1. Bagaimana cara mengkonstruksi model *Convolutional Neural Networks* sebagai model pengenalan wajah manusia dari berbagai sisi?
2. Bagaimana cara memberikan inisialisasi parameter jaringan yang sesuai untuk mengurangi kesalahan dalam proses *training* model *Convolutional Neural Networks* untuk pengenalan wajah?
3. Bagaimana cara mengimplementasikan model *Convolutional Neural Networks* untuk mengenali wajah manusia secara *real-time*?

1.3 Batasan Masalah

Permasalahan dalam dunia nyata sangatlah luas oleh karena itu diberikan suatu batasan masalah pada Tugas Akhir ini antara lain:

1. Perangkat keras Web Cam M-Tech 5MP digunakan sebagai sensor pengenalan wajah secara *real-time*.
2. Pencahayaan pengambilan gambar sebagai dataset dan perekaman pada waktu uji coba diatur dengan kurang pencahayaan dan pencahayaan terang.
3. Definisi *real-time* pada Tugas Akhir ini adalah *real-time* pada saat pengujian.
4. Sisi wajah yang dikenali dibatasi pada bagian depan, samping kiri, samping kanan wajah.

1.4 Tujuan

Adapun tujuan dalam Tugas Akhir ini adalah mengkonstruksi dan mengimplementasikan model *Convolutional Neural Networks* untuk mengenali wajah manusia dari berbagai sisi secara *real-time*.

1.5 Manfaat

Manfaat yang didapatkan Tugas akhir ini adalah sebagai berikut:

1. Dapat digunakan untuk pengenalan identitas seseorang lewat pengenalan wajah secara *real-time* menggunakan *Convolutional Neural Networks* dengan asumsi dataset identitas wajah sudah dilakukan *training* oleh pengguna dan dapat dimanfaatkan oleh pihak instansi atau lembaga seperti pemerintah, kantor polisi, atau universitas untuk tujuan tertentu.
2. Memberikan kontribusi bagi dunia penelitian khususnya dalam pengembangan pengenalan wajah manusia.

1.6 Sistematika Penulisan Tugas Akhir

Tugas Akhir memiliki susunan dalam penulisan agar tersusun secara sistematis dan memudahkan pembaca untuk mempelajarinya. Berikut sistematika penulisan Tugas Akhir ini:

1. BAB I PENDAHULUAN
   Bab ini berisi tentang gambaran umum dari penulisan Tugas Akhir yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA
   Bab ini berisi tentang teori dasar yang mendukung dalam Tugas Akhir ini, antara lain operasi konvolusi, gambar digital, *Extended Local Binary Pattern*, model *Convolutional Neural Networks*, dan analisis jaringan.

3. BAB III METODOLOGI PENELITIAN
   Bab ini menjelaskan tahap pengerjaan dalam menyelesaikan Tugas Akhir ini sehingga penelitian ini dapat dirancang sistematis dan diatur dengan sebaik-baiknya.

4. BAB IV DESAIN SISTEM
   Bab ini menjelaskan tahap persiapan pengolahan data, pengambilan parameter, dan pembuatan model *Convolutional Neural Networks* sebagai acuan dalam implementasi sistem.

5. BAB V IMPLEMENTASI SISTEM
   Bab ini membahas proses untuk implementasi dengan menggunakan bahasa pemograman Java berdasarkan desain sistem yang telah dibuat pada bab sebelumnya.

6. BAB VI UJI COBA DAN EVALUASI SISTEM
   Bab ini membahas tentang pengujian sistem yang telah terimplementasi dengan melakukan proses verifikasi dan validasi beserta pengujian kinerja dari sistem yang telah dibuat.

7. BAB VII KESIMPULAN DAN SARAN
   Bab ini berisi kesimpulan Tugas Akhir yang diperoleh dari bab uji coba dan evaluasi serta saran untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini diuraikan mengenai dasar teori yang digunakan dalam penyusunan Tugas Akhir ini. Dasar teori yang dijelaskan dibagi menjadi beberapa sub bab yaitu penelitian terdahulu, Citra Digital, *Extended Local Binary Pattern,* Operasi Konvolusi, model *Convolutional Neural Networks*, dan analisis jaringan.

2.1 Penelitian Terdahulu

Pada umumnya, metode ekstraksi fitur gambar dua dimensi dalam representasi gambar khususnya wajah secara luas terdapat dua pendekatan umum untuk pengenalan wajah yaitu metode holistik bebasis *interface* dan metode lokal berbasis ekstraksi fitur.

Metode holistik yaitu metode dengan masukan baku berupa daerah keseluruhan wajah sebagai sistem pengenalan. Penelitian awal yang paling sering menggunakan kategori metode ini adalah pengenalan wajah dengan metode dari hasil penerapan Aljabar Linier yaitu *Principal Components Analysis* (PCA)[3]. Metode PCA menggunakan *eigenface* sebagai pengekstrak informasi citra wajah dengan cara mendapatkan kumpulan *eigenvectors* dan matriks kovarian untuk merepresentasikan citra wajah. Pada dasaranya PCA bertujuan untuk menyederhanakan variabel yang diamati dengan cara mereduksi dimensinya. Penelitian lainnya adalah *Liniear Discriminant Analysis*(LDA), secara luas digunakan untuk menemukan kombinasi linear dari fitur dengan tetap mempertahankan pembagian kelas. Tidak seperti PCA, LDA mencoba untuk memodelkan perbedaan antara kelas. Pada awalanya LDA klasik dirancang untuk memperhitungkan dua kelas yang membutuhkan poin data untuk kelas yang berbeda untuk memperpanjang jarak satu sama lain, sementara poin dari kelas yang sama jaraknya diperdekat. Akibatnya, LDA memperoleh perbedaan vektor proyeksi untuk masing-masing

kelas. Tetapi yang lebih sering digunakan adalah algoritma LDA Multi-kelas yang dapat mengelola lebih dari dua kelas [4]. Selanjutnya adalah metode *Independent Component Analysis*(ICA). Penerapan ICA terutama digunakan untuk mencari komponen-komponen independen dari wajah sedemikian sehingga suatu wajah tersebut dapat dinyatakan sebagai kombinasi linear dari komponen-komponen independen yang telah ditemukan. Metode lain berikutnya adalah *Support Vector Machine*, mengklasifikasikan data dengan cara mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas pada ruang input[5]. Secara keseluruhan metode holistik yang dijelaskan sebelumnya dilakukan dengan teknik pengurangan dimensi secara linier yang membuat proses pengenalan di bawah perubahan intentistas cahaya sering mengalami kesalahan.

Metode lokal yaitu metode ekstraksi fitur dari beberapa daerah wajah seperti mulut, mata, dan hidung yang kemudian digunakan untuk klasifikasi. Beberapa metode lokal seperti *Local Binary Pattern* merupakan metode ekstraksi fitur dengan cara mendapatkan nilai biner piksel pada pusat citra dengan 8 nilai piksel disekelilingnya sehingga didapatkan suatu nilai matrik baru yang akan dirubah kesuatu histogram untuk memperoleh fitur vektor wajah[6]. Contoh lain metode lokal lainnya seperti *Scale Invariant Feature Tansform* (SIFT) merupakan metode ekstraksi fitur dengan cara mencari *keypoint* yaitu titik maksimum dan minimum lokal dari hasil perhitungan *Difference of Gaussian*[7]. Kemudian *keypoint* diberikan orientasi yang tetap sehingga tidak terpengaruh pada rotasi citra. *Keypoint* ini yang selanjutnya menjadi fitur-fitur lokal pada suatu citra dan akan dicocokkan dengan *keypoint-keypoint* yang terdapat pada citra lain. Umumnya, metode dalam kategori lokal memerlukan pilihan parameter empiris seperti besar skala dan orientasi filter. Selain itu, perancang harus menentukan daerah yang tepat untuk menerapkan filter yang menjadikan implementasinya optimal. Umumnya, metode dalam kategori ini memerlukan pemilihan

parameter secara empiris seperti jumlah skala dan orientasi dari filter. Selain itu, juga harus menentukan daerah yang tepat untuk penerapannya.

Dalam tugas akhir ini, penulis mengusulkan model *Convolutional Neural Networks*(CNN) dengan input model dari hasil ekstraksi *Extended Local Binary Pattern*(ELBP) sebagai sistem pengenalan wajah. CNN berada pada kategori ekstraksi fitur akan tetapi tetapi memiliki perbedaan dengan kategori metode lokal pada umumnya yaitu posisi dan nilai-nilai dari filter secara otomatis diputuskan oleh CNN selama proses pelatihan tidak langsung manual ditentukan sebelumnya. Kinerja dengan menggunakan model CNN telah dibuktikan oleh berbagai macam aplikasi seperti face detection [8], pengenalan jenis kelamin [9], pengenalan obyek [10], dsb mendapatkan kinerja yang lebih baik dibandingkan metode yang pernah ada sebelumnya. Arsitektur CNN memiliki keunikan dibandingkan metode lainnya antara lain terdapatnya segmentasi, ekstraksi fitur dan klasifikasi dalam satu modul pengolahan. Selanjutnya proses *pre-processing* digunakan metode *Extended Local Binary Pattern* akan mengurangi pengaruh perbedaan intensitas cahaya pada gambar.

2.2 Citra Digital

Citra adalah representasi, kemiripan atau imitasi dari suatu objek atau benda[11]. Secara matematis, citra dinyatakan sebagai suatu fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Citra yang terlihat merupakan cahaya yang direfleksikan dari sebuah objek. Citra dibedakan menjadi dua yaitu citra kontinu diperoleh dari sistem optik yang menerima sinyal analog(mata manusia dan kamera analog) dan citra diskrit(digital) dihasilkan melalui proses digitalisasi terhadap citra kontinu.

Proses digitalisasi pada citra digital dibagi menjadi dua proses yakni sampling dan kuantisaisi. Proses sampling merupakan proses pengambilan nilai diskrit koordinat ruang(x,y) secara periodik dengan periode sampling T. Proses kuantisasi merupakan proses pengelompokkan nilai tingkat keabuan citra

kontinu kedalam beberapa level atau merupakan proses membagi skala keabuan (0,L) menjadi G buah level yang dinyatakan dengan suatu harga bilangan bulat(*integer*), dinyatakan sebagai $G = 2^m$, dengan G adalah derajat keabuan dan m adalah bilangan bulat positif.

Dengan demikian citra digital dapat didefinisikan suatu matriks A berukuran M (baris) x N (kolom) dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya menyatakan tingkat keabuan pada titik tersebut.

$$A = \begin{bmatrix} a_{0,0} & \dots & a_{0,N-1} \\ a_{1,0} & \dots & a_{1,N-1} \\ \vdots & \vdots & \vdots \\ a_{M-1,0} & \cdots & a_{M-1,N-1} \end{bmatrix} \quad (2.1)$$

2.3 *Extended Local Binary Pattern*(ELBP)

Standar operator *Local Binary Pattern(LBP)* didefinisikan sebagai perbandingan nilai biner piksel pada pusat citra dengan 8 nilai piksel disekelilingnya. Variasi tekstur gambar yang disebabkan adanya perbedaaan jumlah pencahayaan berpengaruh pada tingkat akurasi pengenalan. Metode LBP merupakan metode yang efisien untuk menganalisa texture pada gambar. Fitur lokal akan mejadi lebih toleran terhadap perubahan intesitas pencahayaan, distorsi perspektif, dan gambar blur. Metode ini berjalan dengan cara membagi citra kedalam beberapa sel. Setiap pusat piksel didalam sel, dibandingkan dengan piksel $neighbors$ yaitu mengurangkan nilai piksel pada pusat sel dengan nilai piksel disekelilingnya. Tentukan $threshold$ sebagai batasan sehingga jika hasilnya lebih atau sama dengan $threshold$ maka diberi nilai 1 dan jika hasilnya kurang dari $threshold$ maka diberi nilai 0.

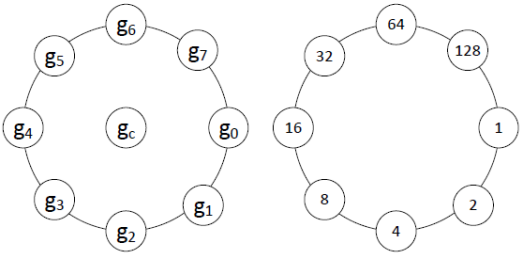
Setelah itu, menyusun deretan 8 bit biner searah jarum jam atau sebaliknya dan merubah 8 bit biner kedalam nilai desimal untuk menggantikan nilai piksel pada pusat citra[12]. Secara umum LBP didefinisikan sebagai berikut:

$$s(g_p - g_c) = \begin{cases} 1, & g_p \geq g_c \\ 0, & g_p < g_c \end{cases} \quad (2.2)$$

$$LBP_{R,P} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \quad (2.3)$$

P : jumlah *neighbour*
R : radius pada *circular neighbour*
g_c : nilai intensitas pusat piksel (*threshold*)
g_p : nilai intensitas *neighbour*
p : indeks *neighbour*

Pada **Gambar 2.1** adalah contoh LBP dengan $P = 8$ beserta bobot yang berkorespondensi dengan *neighbour*. Jika koordinat pada pusat piksel adalah (x, y) maka koordinat jarak seragam *circular neighbouhood* diberikan sebagai $(x + Rcos\left(\frac{2\pi p}{P}\right), y - Rsin\left(\frac{2\pi p}{P}\right))$ untuk $p = 0,1,\ldots,P-1$. Jika koordinat *neighboor* tidak berkorespondensi ke bilangan bulat, maka digunakan interpolasi bilinier sebagai estimasi pada nilai piksel.



(a) (b)

Gambar 2.1. (a) Neighboorhood pada LBP (b) Bobot yang terkait dengan neighbour

Perluasan dari LBP yaitu *Extended Local Binary Pattern*(ELBP) atau *Rotated Local Binary Pattern*(RLBP). Perluasan ini dalam rangka untuk membuat LBP invarian terhadap rotasi maka

dilakukan penggeseran bobot secara sirkular sesuai dengan arah domain. Arah domain(D) dalam *neighbourhood* adalah indeks pada *neighbour* yang memiliki perbedaan maksimal dengan nilai intensitas pusat piksel. Didefinisikan sebagai berikut:

$$D = \underset{p \in (0,1 \ldots P-1)}{\arg max} |g_p - g_c| \quad (2.4)$$

Rotasi *neighbourhood* dengan pusatnya bergeser ke arah *D* dengan sudut yang sama. Sebagai contoh *neigbourhood* [23 25 28; 167 35 31; 56 67 72] setelah dilakukan *thresholding* kode biner menjadi [0111**1**000], indeks *D* adalah 4 ditunjukkan pada tulisan bercetak tebal pada angka biner, berkorespondensi ke nilai piksel 167. Jika gambar diputar 45^oberlawanan arah jarum jam, maka *neighboorhood* juga berputat dengan sudut yang sama dan kode biner bergeser menjadi [00111**1**00]. Indeks *D* tetap berkorespondensi dengan nilai yang sama 167 tetapi bergeser satu langkah. Oleh karena arah dominan diambil sebagai acuan dalam *circular neighbourhood*, maka bobot juga berdasarkan acuan tersebut. Sehingga operator RLBP didefinisikan sebagai berikut:

$$RLBP_{R,P} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^{mod(p-D,P)} \quad (2.5)$$

Dimana mod mengindikasikan operasi modulo. Pada definisi bobot $2^{mod(p-D,P)}$ bergantung pada D. Bobot yang sirkuler bergeser sehubungan dengan arah dominan. Hasil pergeseran invarian terhadap rotasi, karena bobot sekarang tergantung pada *neighbourhood* dan bukan pada susunan yang dipilih sebelumnya.

2.4 Operasi Konvolusi

Di dalam matematika, konvolusi adalah sebuah operasi matematika dari dua fungsi f dan g yang menghasilkan fungsi ketiga h. Jenis konvolusi terdapat dua jenis yaitu konvolusi kontinu dan konvolusi diskrit. Didalam pengolahan citra diberlakukan pada domain spasial sehingga konvolusi yang

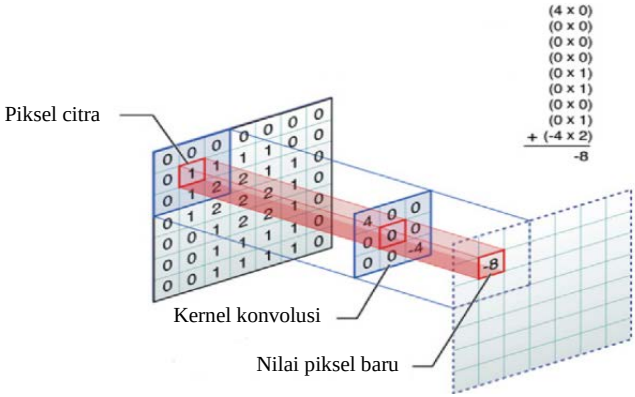
digunakan adalah jenis konvolusi diskrit dengan operasinya secara umum didefinisikan sebagai berikut:

$$h(x,y) \coloneqq (f * g)(x,y) \coloneqq \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a,b)g(x-a,y-b) \qquad (2.6)$$

Fungsi $f(x,y)$ adalah sebuah fungsi yang direpresentasikan sebagai citra dan fungsi $g(x,y)$ adalah kernel konvolusi atau kernel penapis *(filter)*. Kernel $g(x,y)$ merupakan suatu jendela yang dioperasikan secara bergeser pada sinyal masukan $f(x,y)$, yang dalam hal ini, jumlah perkalian kedua fungsi pada setiap titik merupakan hasil konvolusi dinyatakan sebagai keluaran fungsi $h(x)$. Oleh karena ukuran citra terbatas pada ukuran MxN dengan kernel berukuran mxn maka operasi fungsi konvolusi didefinisikan sebagai berikut[11]:

$$(f * g)(x,y) \coloneqq \sum_{a=x-h}^{x+h} \sum_{b=y-w}^{y+h} f(a,b)g(x-a,y-b) \qquad (2.7)$$

Dimana $m = 2h + 1$ adalah tinggi kernel dan $n = 2w + 1$ adalah lebar kernel. Operasi dilakukan dengan melakukan kombinasi linear dari mengambil bagian input citra yang sama dengan kernel dan nilai hasil operasi disimpan berupa elemen nilai matriks kemudian dilanjutkan dengan menggeser kernel piksel per piksel sampai terhimpun keseluruhan nilai piksel baru[13]. Proses konvolusi disajikan pada **Gambar 2.2** dengan citra berukuran $M = 7$ $N = 7$ dan kernel berukuran $m = 3$ $n = 3$. Operasi pertama dilakukan dengan mengalikan bagian dari elemen citra pertama berukuran 3x3 (sesuai dengan ukuran kernel) dengan tiap elemen kernel yang bersesuaian kemudian dilakukan penjumlahan pada akhir perhitungannya.



Gambar 2.2. Proses operasi konvolusi pada satu lokasi pada suatu citra

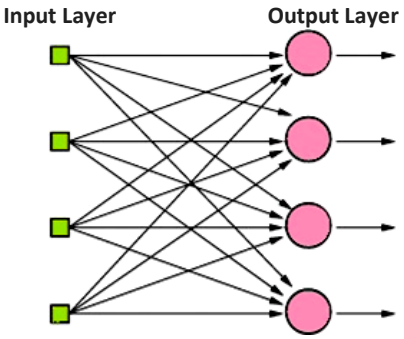
2.5 Convolutional Neural Networks(CNN)

CNN adalah variasi dari *Multilayer Perceptron* yang terinspirasi dari jaringan syaraf manusia. Penelitian awal yang mendasari penemuan ini pertama kali dilakukan oleh Hubel dan Wiesel[14] yang melakukan penelitian *visual cortex* pada indera penglihatan kucing. *Visual cortex* adalah bagian dari otak yang berfungsi untuk memproses informasi visual yang didalamnya berisi susunan kompleks dari sel. Sel-sel ini sensitif terhadap bagian daerah kecil*(sub-regions)* pada bidang visual disebut sebagai bidang reseptif*(receptive field)*. Daerah bagian berbentuk seperti ubin berfungsi untuk menutupi seluruh bidang visual. Sel-sel ini bertindak sebagai *filter* lokal atas ruang input dan dapat dieksploitasi pada daerah spasial lokal gambar. Selain itu dari hasil penelitian terindetifikasi 2 tipe sel dasar, sel tipe sederhana merespon maksimal terhadap pola tertentu seperti tepi didalam bidang reseptifnya sedangkan sel tipe kompleks memiliki bidang reseptif yang lebih besar dan lokal invariant terhadap posisi yang sesuai pada pola. *Visual cortex* pada hewan sangat *powerful* dalam sistem pemrosesan visual yang pernah ada. Hingga banyak penelitian yang terinspirasi dari cara kerjanya dan menghasilkan

model-model baru diantaranya seperti Neocognitron [15], HMAX [16], dan LeNet-5 [3].

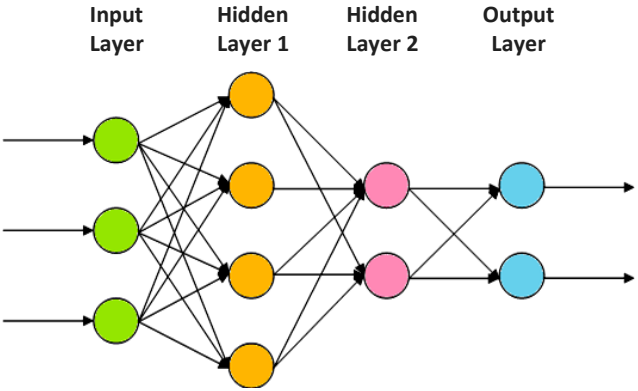
2.5.1 Gambaran Umum Arsitektur

Neural Networks menerima input berupa vektor tunggal dan ditransformasikan sepanjang *hidden layer*. Pada *Multilayer Neural Netwroks*, *s*etiap *hidden layer* tersusun dari kumpulan *neuron*, dimana setiap *neuron* terhubung secara penuh*(fully connected)* ke semua *neuron* pada *layer* sebelumnya, berbeda dengan *Single Layer Neural Networks* yang hanya ada neuron yang terhubung secara penuh pada layer terakhir tanpa adanya pembagian koneksi antar neuron*(no hidden layer)*. *Layer* terakhir pada keseluruhan jaringan bersifat terhubung secara penuh dan merupakan representasi dari nilai kelas klasifikasi.



Single Layer Feedforward Network

(a)



Multi- Layer Feedforward Network

(b)

Gambar 2.3. (a) Single Neural Networks tanpa hidden layer (b) Multi Neural Networks dengan sebuah hidden layer

Bilamana *Neural Networks* diberikan input berupa gambar dengan dimensi MxNxC(M lebar, N tinggi, C *channel*) dihubungkan secara penuh maka jumlah parameter bobot sama dengan MxNxC bobot merupakan jumlah parameter yang sangat besar ketika ukuran gambar sangat besar. Dihubungkan secara penuh kurang efektif ketika dihubungkan dengan banyak parameter yang nantinya dapat menyebabkan *overfitting*. *Overfitting* merupakan kondisi sistem kehilangan sifat generalnya atau hanya dapat mengenali *dataset training* dimana biasanya disebabkan karena arsitektur jaringan terlalu dalam dan disebabkan juga karena sistem terlalu banyak melakukan iterasi.

Convolutional Neural Networks(CNN) didesain untuk mengatasi permasalahan ini. *Convolutional Neural Networks* merupakan suatu *layer* yang memiliki susunan *neuron* 3D (lebar, tinggi, kedalaman). Lebar dan tinggi merupakan ukuran *layer* sedangkan kedalaman mengacu pada jumlah *layer*. Secara umum jenis *layer* pada CNN dibedakan menjadi dua yaitu:

a. *Layer* ekstraksi fitur gambar, letaknya berada pada awal arsitektur tersusun atas beberapa *layer* dan setiap *layer* tersusun atas *neuron* yang terkoneksi pada daerah lokal *(local region) layer* sebelumnya. *Layer* jenis pertama adalah *layer* konvolusi dan *layer* kedua adalah *layer pooling*. Setiap *layer* diberlakukan fungsi aktivasi. Posisinya berselang-seling antara jenis pertama dengan jenis kedua. *Layer* ini menerima input gambar secara langsung dan memprosesnya hingga menghasilkan keluaran berupa vektor untuk diolah pada layer berikutnya.
b. *Layer* klasifikasi, tersusun atas beberapa layer dan setiap *layer* tersusun atas *neuron* yang terkoneksi secara penuh *(fully connected)* dengan *layer* lainnya. *Layer* ini menerima input dari hasil keluaran *layer* ekstrasi fitur gambar berupa vektor kemudian ditransformasikan seperti *Multi Neural Networks* dengan tambahan beberapa *hidden layer*. Hasil keluaran berupa skoring kelas untuk klasifikasi.

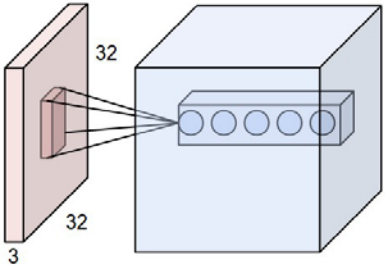
Dengan demikian CNN merupakan metode untuk mentransformasikan gambar original *layer* per *layer* dari nilai piksel gambar kedalam nilai skoring kelas untuk klasifikasi. Dan setiap *layer* ada yang memiliki *hyperparameter* dan ada yang tidak memiliki *parameter*(bobot dan bias pada *neuron*).

2.5.2 *Convolutional Layer*

Layer yang pertama kali menerima input gambar langsung pada arsitektur. Operasi pada *layer* ini sama dengan operasi konvolusi yaitu melakukan operasi kombinasi linier *filter* terhadap daerah lokal. *Filter* merupakan representasi bidang reseptif dari *neuron* yang terhubung kedalam kedalam daerah lokal*(local connectivity)* pada input gambar. Bentuk *layer* direpresentasikan sebagai volume BxKxL atau *layer* ukuran BxK dengan jumlah sebanyak L. *Convolutional layer* memiliki *hyperparameter* dan parameter. *Hyperparameter* pada *layer* ini menjadi acuan untuk menentukan jumlah dan ukuran hasil ekstraksi *layer*, dapat dilihat pada **Tabel 2.1**.

Tabel 2.1. *Hyperparameter* pada Convolutional Layer

No	*Hyperparameter*	Keterangan
1	*Depth*	Kedalaman *layer* atau jumlah *layer* konvolusi
2	*Stride*	Jumlah pergeseran *filter* pada proses konvolusi
3	*Zero-padding*	Jumlah penambahan nilai intensitas nol di daerah sekitar input gambar



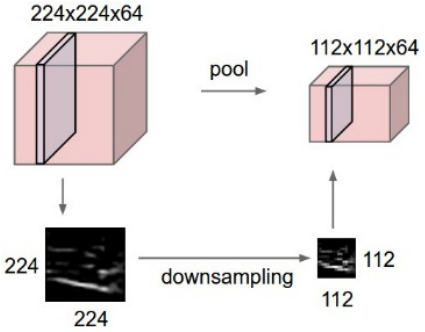
Gambar 2.4. Input gambar berukuran 32x32x3 dan volume *convolutional layer* pada *layer* pertama. Dengan setiap *neuron* terhubung pada daerah lokal input

Parameter sharing pada *convolutional layer* digunakan untuk mengotrol jumlah parameter. Parameter bagian *layer* ini merupakan himpunan *learnable filter* atau suatu kernel yang nilainya berupa bobot. Jika meninjau dari arsitektur NN, ukuran gambar adalah MxNxC sebanyak N maka jumlah neuron adalah MxNxCxN neuron dan *filter* berukuran BxK sebanyak L maka jumlah neuron adalah BxKxL sehingga total keseluruhan parameter adalah MxNxCxNxBxKxL. Hal tersebut berlaku hanya bila kondisi parameter dibuat berbeda pada setiap input piksel gambar. Jumlah parameter ini sangat tidak efisien dan memboroskan ruang *memory*. Oleh karena itu model *Artificial Neural Networks* tidak digunakan. Pada *ConvLayer* setiap parameter disamakan pada setiap piksel input gambar sehingga

jumlah parameter berubah menjadi sebanyak MxNxCxBxKxL parameter.

2.5.3 *Pooling Layer*

Pooling layer akan mereduksi ukuran spasial dan jumlah parameter dalam jaringan serta mempercepat komputasi dan mengontrol terjadinya *overfitting*.



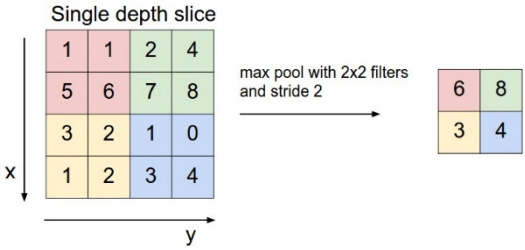
Gambar 2.5. Representasi Pooling Layer

Pooling layer bekerja dengan blok spasial yang bergerak sepanjang ukuran *feature pattern*. Ukuran pergeseran blok pada umumnya adalah ukuran pada dimensi blok (HxH) itu sendiri sehingga tidak ada *overlapping* seperti pada *Convolutional Layer*. Pergerakan blok diikuti dengan perhitungan *pooling* pada masukan pola fitur (u). Pada *layer* ini tidak memiliki parameter karena parameter sudah ditentukan sebelumnya*(fixed)*. *Pooling layer* memiliki beberapa macam tipe antara lain sebgai berikut:

a. Max Pooling

   Keluaran pola fitur (z) didapatkan dengan mencari nilai yang paling besar pada blok spasial pada setiap pergerakan. Metode ini merupakan metode standar yang digunakan dalam penelitian.

$$u_{i,j}^{(k)} = \max_{p,q \in P_{i,j}} z_{p,q}^{(k)} \tag{2.8}$$



Gambar 2.6. Perhitungan max pooling dengan *stride* 2

b. Average Pooling
Mendapatkan keluaran pola fitur dengan menghitung nilai rata-rata masukan pola fitur pada blok spasial HxH piksel di setiap area pergerakan. Average pooling tetap menyimpan informasi spasial pada pola fitur sebelumnya.

$$u_{i,j}^{(k)} = \frac{1}{H^2} \sum_{(p,q)\in P_{ij}} z_{p,q}^{(k)} \tag{2.9}$$

c. Lp Pooling
Adalah tipe pooling layer generalisasi dari Max pooling dan Average pooling.

$$u_{i,j}^{(k)} = \left(\frac{1}{H^2} \sum_{(p,q)\in P_{ij}} \left(z_{p,q}^{(k)} \right)^P \right)^{\frac{1}{P}} \tag{2.10}$$

Ketika nilai $P = 1$, pooling layer bekerja sebagai average pooling dan jika nilai $P = \infty$ pooling layer bekerja sebagai Max Pooling.

2.5.4 Fungsi Aktivasi*(Neurons)*

Fungsi aktivasi atau fungsi transfer merupakan fungsi non-liniear yang memungkinkan sebuah jaringan untuk dapat

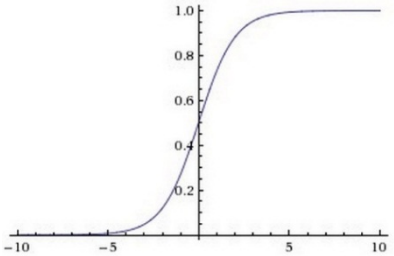
menyelesaikan permasalahan permasalahan non trivial. Setiap fungsi aktivasi mengambil sebuah nilai dan melakukan operasi matematika. Pada arsitektur CNN, fungsi aktivasi terletak pada perhitungan akhir keluaran *feature map* atau sesudah proses perhitungan konvolusi atau pooling untuk menghasilkan suatu pola fitur. Beberapa macam fungsi aktivasi yang sering digunakan dalam penelitian adalah sebagai berikut:

a. Fungsi sigmoid

Dalam matematika fungsi sigmoid memiliki definisi persamaan fungsi sebagai berikut[17]:

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{2.11}$$

Fungsi ini memiliki keluaran dengan rentang[0,1] dengan demikian fungsi ini mengkonversi sebuah nilai input ke dalam kisaran 0 sampai 1. Jika input bernilai sangat negatif maka fungsi akan menghasilkan angka nol dan jika input bernilai sangat positif maka fungsi akan menghasilkan hasil angka 1. Fungsi sigmoid jarang digunakan karena keluaran fungsi tidak terpusat pada nol dan dapat membuat jenuh *training* saat $f(x) = 0\ atau\ 1$ sehingga dapat menyebabkan gradien sama dengan nol.



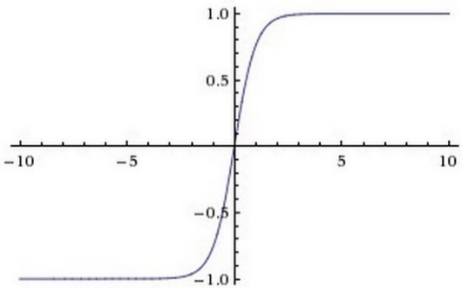
Gambar 2.7. Grafik fungsi sigmoid biner

b. Fungsi Tanh

Definisi persamaan fungsi tanh atau fungsi sigmoid bipolar adalah sebagai berikut[18]:

$$f(x) = \tanh(x) \tag{2.12}$$

Fungsi tanh memiliki keluaran dengan rentang [-1,1]. Fungsi ini merupakan fungsi aktivasi yang berbeda dengan fungsi aktivasi sigmoid sebelumnya, keluaran fungsi ini bersifat terpusat pada nol sehingga dan lebih cepat konvergen dibandingkan fungsi sigmoid sehingga dalam aplikasinya paling sering digunakan.



Gambar 2.8. Grafik fungsi sigmoid bipolar

c. Fungsi Rectified Liniear Unit(ReLU)

Definisi persamaan fungsi *(Rectifiers Liniear Unit)*ReLU adalah sebagai berikut[19]:

$$f(x_i) = \begin{cases} x_i, & x_i \geq \epsilon \\ 0, & x_i < \epsilon \end{cases} \quad (2.13)$$

Dimana x_i adalah input dari fungsi aktivasi *non-liniear* pada *channel* ke-i. Fungsi ReLU digunakan pada tahap *forward pass* CNN adalah $y = \max(\epsilon, x)$. Sedangkan untuk tahap *backward pass* digunakan fungsi turanannya yaitu:

$$f(x_i) = \begin{cases} 1, & x_i \geq \epsilon \\ 0, & x_i < \epsilon \end{cases} \quad (2.14)$$

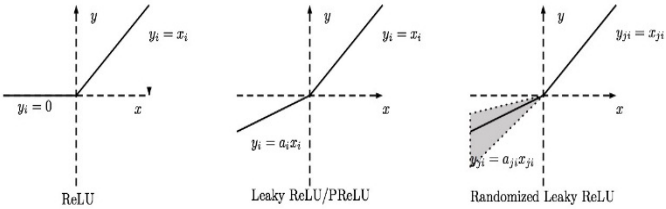
Walaupun ReLU tidak tediferensiasi di ϵ. Tetapi mempunyai subdifferensial pada selang [0,1]. Sembarang nilai pada subinterval dapat diterima sebagai subdifferensiasi sehingga dapat digunkan dalam SGD jika digeneralisasi dari *gradient descent*.

d. Fungsi Leaky ReLU(LReLU)

Definisi persamaan fungsi Leaky ReLU adalah sebagai berikut[20]:

$$f(x_i) = \begin{cases} x_i, & x_i \geq \epsilon \\ 0.01\, x_i, & x_i < \epsilon \end{cases} \quad (2.15)$$

Fungsi aktivasi LReLU bertujuan untuk menghindari nilai gradient nol. Dibandingkan dengan fungsi sigmoid dan tanh, funsi ReLU lebih mudah dan cepat dalam perhitungannya karena dapat diimplementasikan dengan memeberikan suatu *threshod* dan memiliki akselerasi yang tinggi pada pencarian kekokvergenitas. Namun fungsi ini juga memiliki kelemahan



Gambar 2.9. Grafik fungsi ReLU(kiri), fungsi LReLU(tengah), dan fungsi PReLU(kanan)

e. Fungsi *Parametric ReLU*

Definisi persamaan fungsi Parameteric ReLU adalah sebagai berikut[21]:

$$f(x_i) = \begin{cases} x_i, & x_i \geq \epsilon \\ a_i x_i, & x_i < \epsilon \end{cases} \quad (4.16)$$

Dimana x_i adalah input dari fungsi aktivasi *non-liniear* pada *channel* ke-i, dan a_i adalah koefisien kontrolling kemiringan pada sumbu negatif. Index i pada a_i sebagai indikasi bahwa nilai koefisien kontrolling dapat berbeda pada setiap *channel*.

Fungsi ini merupakan generalisasi dari fungsi *rectified* yaitu ketika $a_i = 0$ maka disebut sebagai fungsi ReLU*(Rectifiers Liniear Unit)*. Jika nilai a_i bernilai kecil dan sudah ditetapkan sama dengan 0.01 maka disebut sebagai LReLU*(Leaky ReLU)*.

2.6 Library OpenCV

OpenCV adalah sebuah library fungsi pemrrograman yang ditujukan untuk *computer vision*. Awalnya dikembangkan oleh pusat penelitian Intel di Nizhny Novgorod (Rusia), kemudian didukung oleh Willow Garage dan sekarang dikelola oleh Itseez. Library OpenCV di bawah lisensi BSD *open-source* gratis dan *cross-platform* untuk digunakan. Didalamnya terdapat ratusan algoritma computer vision.

OpenCV memiliki struktur modul, dengan modulnya sebagai berikut:

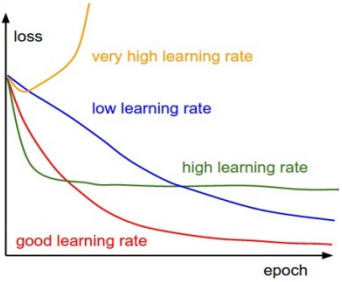
- *Core functionality* - mendefinisikan struktur data dasar, termasuk multi-dimensi array (Mat) dan fungsi dasar yang digunakan oleh semua modul lainnya.
- *Image Processing* - modul pengolahan citra yang mencakup pengolahan gambar linear dan *non-linear filtering*, transformasi gambar geometri (mengubah ukuran, *affine* dan *perspective warping*, *generic table-based remapping*), konversi ruang warna, histogram, dan sebagainya.
- Video - modul analisis video yang mencakup estimasi gerakan, *background substraction*, dan algoritma pelacakan obyek.
- calib3d - dasar algoritma geometri *multiple-view geometry*, kalibrasi kamera tunggal dan stereo, estimasi posisi obyek, algoritma korespondensi stereo, dan unsur-unsur rekonstruksi 3D.
- features2d - *salient feature detectors*, deskripsi, dan *matchers descriptor*.
- objdetect - deteksi objek dan beberapa contoh obyek yang sudah didefinsikan (misalnya, wajah, mata, mulut, orang, mobil, dan sebagainya).
- highgui - antarmuka yang digunakan untuk UI sederhana.
- videoio - antarmuka yang digunakan untuk menangkap video dan *codec* video.
- gpu - algoritma GPU-*accelerated* dari modul OpenCV

- dan beberapa modul pembantu lainnya, seperti FLANN dan *Google test wrappers*, Python *bindings*, dan lain-lain.

2.7 Analisis Jaringan

Verifikasi jaringan agar sesuai kebutuhan sistem ditinjau dari segi mekanis dapat dilakukan dengan analisis konvergensi dan kestabilan jaringan.

Analisis konvergensi dan kestabilan merupakan tahap analisas kesesuaian konstruksi jaringan sesuai dengan kebutuhan yang sudah diidentifikasi sebelumnya. Konvergensi suatu jaringan dapat dilihat dari perubahan nilai *error* pada waktu *training* setiap iterasi. Kestabilan jaringan apat dilihat dari perubahan nilai *error* makin lama makin kecil sehingga nilai *error* mendekati konstan yaitu pada *error minimal*. Jika nilai error berkonvergensi ke sebuah nilai ke *error* minimal(konvergen dan stabil) melalui iterasi-iterasi yang dijalani selama tahap *training* maka jaringan dapat mengklasifikasikan data dengan benar. Bilamana jaringan tidak konvergen ke suatu titik (perubahan nilai *error* tidak mengarah kemanapun) dapat diambil kesimpulan bahwa jaringan tidak dapat melakukan pembelajaran atau tidak dapat mengklasifikasikan data dengan benar. Beberapa kemungkinan jaringan tidak konvergen dapat disebabkan karena keselahan pada tahap desain atau implementasi sistem.



Gambar 2.10. Kurva pengaruh *learning rate* terhadap konvergensi dan kestabilan jaringan

Kestabilan dan kecepatan konvergen jaringan dipengaruhi oleh nilai *learning rate*, dapat dilihat pada **Gambar 2.10**.

1. Jika nilai *learning rate* terlalu tinggi maka nilai *error* akan konvergen ke nilai maksimal *error* atau jaringan gagal dalam proses pembelajaran sehingga tidak stabil.
2. Jika nilai *learning rate* tinggi maka tahap pembelajaran benar akan tetapi mendapatkan *error* yang konstan dan besar(kurang stabil).
3. Jika nilai *learning rate* rendah maka terlalu rendah maka tahap pembelajaran benar akan tetapi membutuhkan waktu yang lama untuk mendapatkan nilai *error* minimal(stabil).

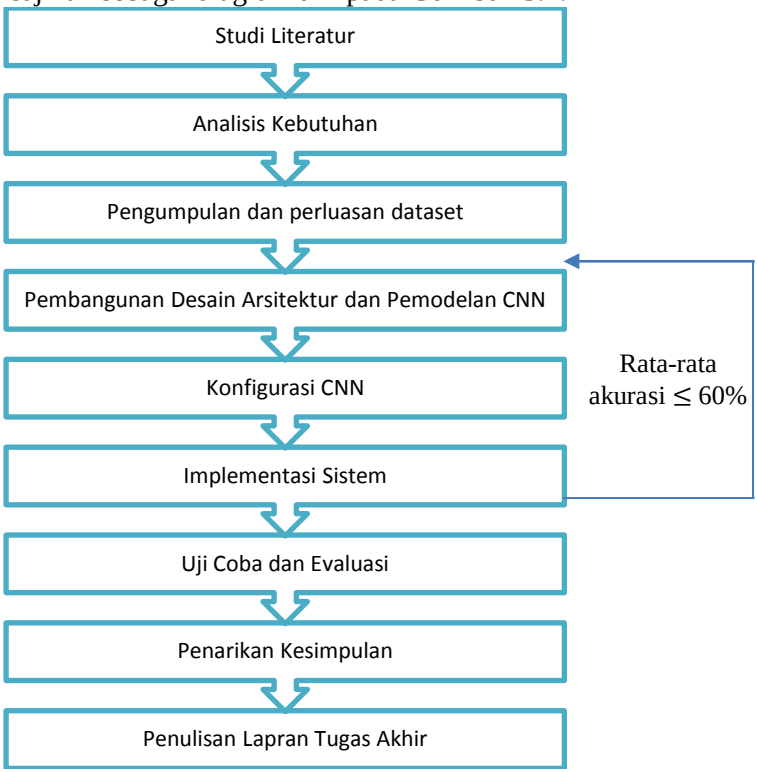
Dengan demikian katergori *nilai learning* yang baik adalah berada diantara kategori rendah dan tinggi. Sehingga jaringan konvergen dan stabil.

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan tentang tahap pengerjaan dalam menyelesaikan Tugas Akhir ini sehingga penelitian ini dapat dirancang sistematis dan diatur dengan sebaik-baiknya.

3.1 Diagram Metodologi

Gambaran tahap-tahap dalam penelitian pada Tugas Akhir ini disajikan sebagai diagram alir pada **Gambar 3.1.**



Gambar 3.1. Diagram alir metodologi penelitian

3.2 Studi Literatur

Tahap pertama adalah tahap mengumpulkan referensi-referensi dan mengkaji terkait pengolahan citra digital, metode *Extended Local Binary Pattern*, model *Convolutional Neural Networks*, *Open Source Computer Vision Library*(OpenCV), beserta penelitian-penelitian pengenalan wajah manusia dari literature jurnal, buku, artikel, dan tugas akhir.

3.3 Analisis Kebutuhan

Tahap ke dua dilakukan analisis kebutuhan sistem sebagai gambaran umum rancangan aplikasi. Berikut daftar kebutuhan sistem yang menjadi acuan pada rancangan aplikasi Tugas Akhir ini:

a. Sistem dapat melakukan perekaman secara *real-time* dengan resolusi video 640x480 piksel.
b. Sistem dapat mendeteksi lebih dari satu wajah dari berbagai posisi sesuai batasan masalah.
c. Sistem dapat mengambil sampel gambar wajah yang teridentifikasi secara *real-time* kemudian disimpan menjadi sebuah *file* gambar *grayscale* berukuran 48x48 piksel untuk *training* jaringan.
d. Sistem memiliki menu *training* sebagai fitur pelatihan gambar wajah seseorang untuk membuat model baru pengenalan wajah seseorang sesuai input *file* gambar yang kemudian model disimpan dalam bentuk sebuah *file* berisi model obyek.
e. Sistem dapat mengenali wajah secara *real-time* dari wajah yang terdeteksi berdasarkan model yang dimasukkan.

3.4 Pengumpulan dan Perluasan Dataset

Dataset berupa himpunan gambar wajah yang dibagi menjadi dua jenis himpunan. Himpunan gambar pertama adalah himpunan gambar wajah *indoor* atau gambar wajah dalam kondisi

pencahayaan minim sedangkan himpunan gambar kedua adalah gambar wajah *outdoor* atau gambar wajah dalam kondisi pencahayaan terang.

3.5 Pembangunan Desain Arsitektur dan Pemodelan CNN

Desain arsitektur dimulai dari menentukan kedalaman jaringan, susunan *layer*, dan pemilihan jenis *layer*. Setiap *layer* ditentukan *parameter* dan *hyperparameter* sebagai tinjaun penyusunan model. Model jaringan dikonstruksi menjadi suatu definisi persamaan fungsi pada tiap *layer*.

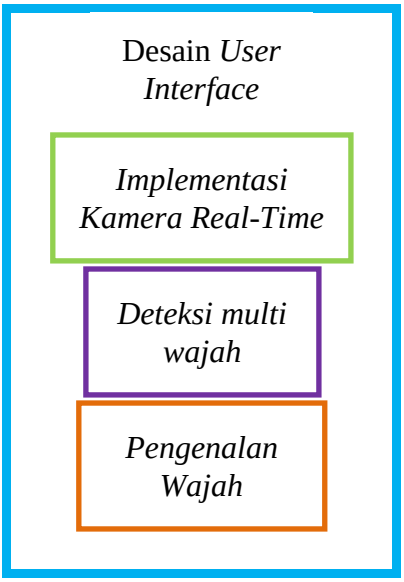
3.6 Konfigurasi CNN

Pengoptimalan kinerja sistem dilakukan dengan menentukan kesesuaian inisialisasi parameter jaringan dan memberikan metode tambahan untuk meningkatkan kinerja jaringan dari studi literetur yang sudah dilakukan sebelumnya.

3.7 Implementasi Sistem

Aplikasi dikembangkan dengan bahasa pemrogaman Java dengan bantuan *Library OpenCV*. *Library OpenCV* menghubungkan webcam dan mengimplementasikan multi deteksi wajah dari berbagai posisi. Masukan sistem berupa *frame* gambar *real-time*. Keluaran sistem berupa posisi wajah dan nama label dari hasil deteksi dan pengenalan. Secara umum proses pendekatan pengenalan wajah terdiri dari tiga langkah yang dapat dilihat pada **Gambar 3.2.** dan susunan desain utama sistem disajikan pada **Gambar 3.3**.

Gambar 3.2. Proses sistem pengenalan wajah

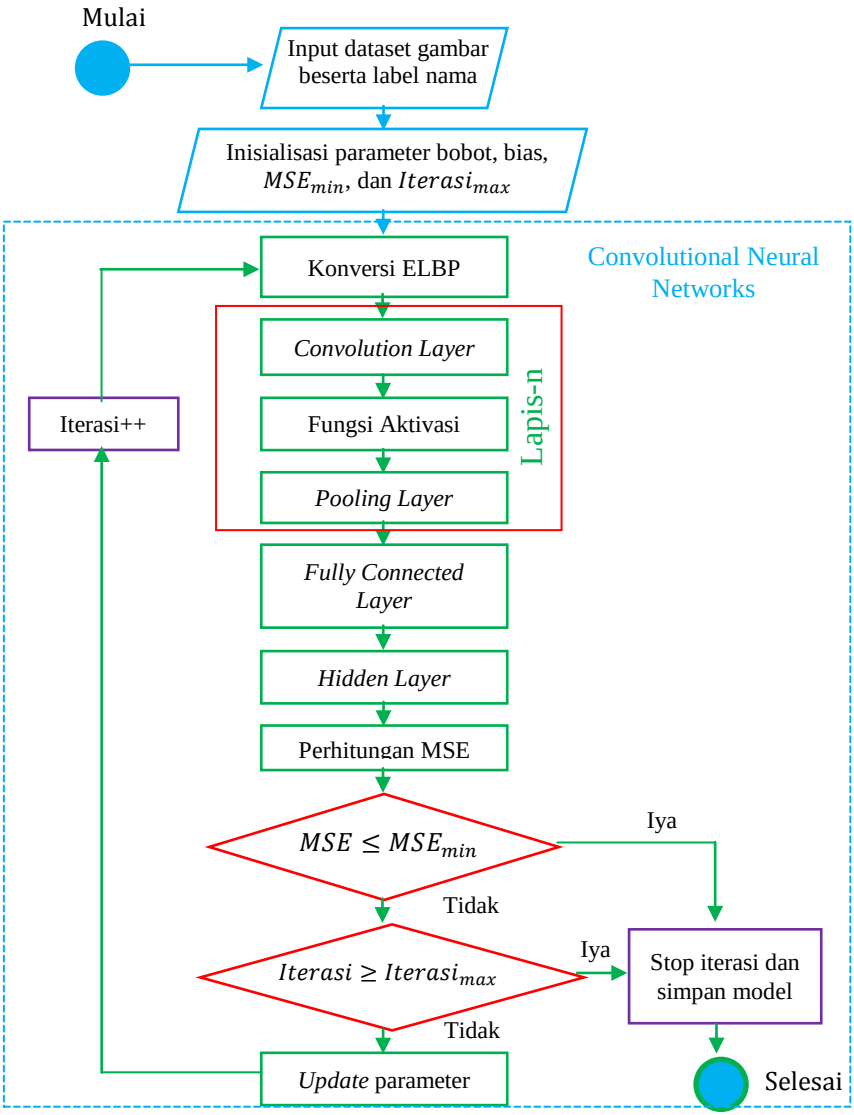


Gambar 3.3. Desain utama implementasi system

3.8 Uji Coba dan Evaluasi

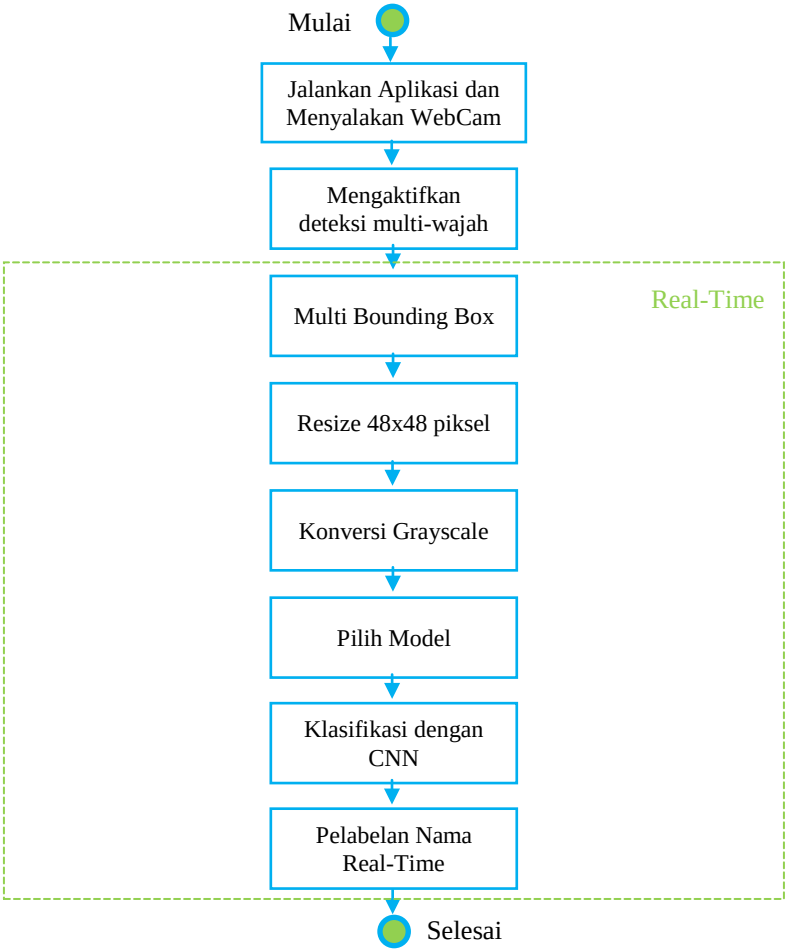
Tahap uji coba aplikasi dilakukan dengan menjalankan implementasi rancangan aplikasi beserta pengujian kesesuaian terhadap kebutuhan sistem.

Dalam proses uji coba aplikasi terdapat dua tahap yaitu uji coba *training* dan uji coba model. Uji coba *training* adalah uji coba jaringan untuk mendapatkan model jaringan berdasarkan input data pelatihan dan nama label disertai analisis pola *error* untuk memvalidasi proses pembuatan model. Adapun diagram alir uji coba *training* disajikan pada **Gambar 3.3**.



Gambar 3.4. Diagram alir uji coba *training Convolutional Neural Networks*

Sedangkan tahap uji coba model adalah tahap uji coba pengenalan wajah secara *real-time* menggunakan hasil keluaran model *training* yang disajikan pada **Gambar 3.4**.



Gambar 3.5. Diagram alir uji coba model

Hasil keluaran model pada perekaman wajah orang secara *real-time* akan dianalisis dengan perhitungan tingkat akurasi sebagai berikut:

$$AC = \frac{n}{N} x100\%$$

dengan AC adalah persentase akurasi model, n adalah jumlah kebenaran pengenalan wajah dan N adalah jumlah wajah yang terdeteksi.

3.9 Penarikan Kesimpulan

Tahap selanjutnya adalah penarikan kesimpulan dari keseluruhan tahap yang sudah dilakukan dan pemberian saran terkait kekurangan hasil penelitian untuk pengembangan pengenalan wajah berikutnya.

3.10 Penulisan Laporan Tugas Akhir

Tahap akhir setelah semua tahap dijalankan adalah penulisan laporan tugas akhir. Konten laporan tugas akhir sesuai dengan hasil yang telah didapatkan dari proses awal pengumpulan data sampai dengan penarikan kesimpulan.

BAB IV
DESAIN SISTEM

Bab ini menjelaskan rancangan desain sistem yang digunakan sebagai acuan untuk implementasi sistem. Desain sistem menggambarkan proses rancang bangun secara terperinci dari awal tahap pengumpulan data hingga pembuatan dan konfigurasi model *Convolutional Neural Networks*.

4.1 Pengumpulan dan Perluasan Dataset

Pengumpulan data gambar dilakukan langsung menggunakan kamera Web Cam. Setiap gambar wajah akan dirubah dan disimpan dalam bentuk *grayscale* berukuran 48x48 piksel untuk dijadikan sebagai *dataset*. Akan tetapi *dataset* yang didapatkan masih sangat terbatas untuk mendapatkan kinerja optimal pada sistem, oleh karena itu dilakukan *data augmentation* untuk menambah variasi *dataset*, dapat dilihat pada **Tabel 4.1.**

Tabel 4.1. Daftar metode *data augmentastion*

1	***Horizontally flipping***	
	Pembalikan gambar wajah secara horizontal	
2	***Translation***	
	Pergeseran gambar wajah dari 1 piksel ke 2 piksel	
3	***Rotation***	
	Perputaran gambar wajah mulai dari 3 sampai 6 derajat	
4	***Scaling***	
	Resize dengan skala antara 0.9 ~ 1.1	

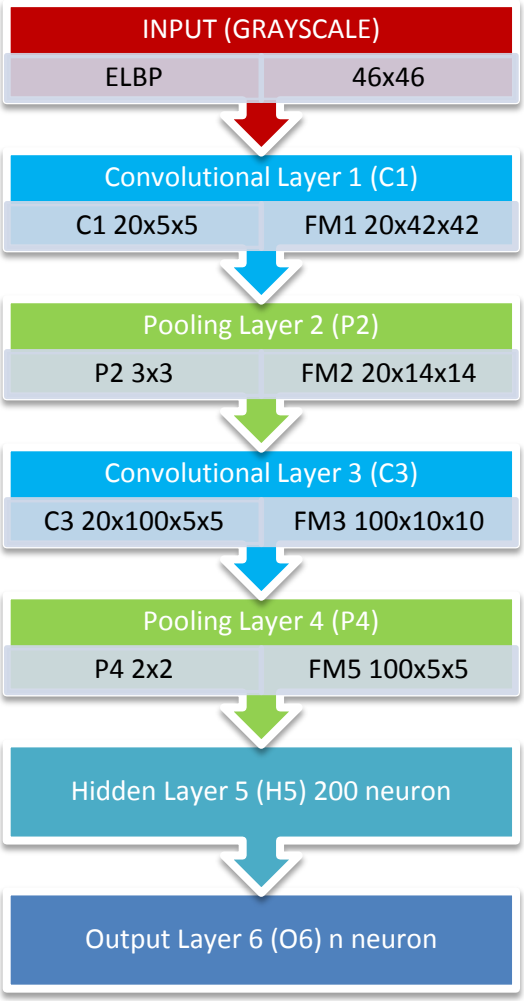
5	***Randomly blackening***	
	Penghitaman secara acak dengan itensitas 20%	

4.2 Data Preprocessing

Gambar 48x48 grayscale diekstraksi terlebih dahulu menggunakan metode *Extended Local Binary Pattern* sebelum masuk jaringan. Metode ini dipilih karena hasil ekstraksi lebih toleran terhadap perubahan intesitas pencahayaan, distorsi perspektif, blur, perubahan skala gambar, dan sebagainya. Pada Tugas Akhir ini pengaturan radius diberi nilai 1 dan *neighbor* diberi nilai 15 sehingga dimensi gambar dataset berubah menjadi 46x46.

4.3 Desain Arsitektur *Convolutional Neural Networks*

Kedalaman arsitektur mempengaruhi tingkat kemampuan jaringan. Arsitektur *Convolutional Neural Networks* dalam Tugas akhir ini dirancang dengan kedalaman 7 *layer* model konvolusi antara lain *input layer, convolutional layer C1, pooling layer P2, convolutional layer C3, pooling layer P4, hidden layer H5* dan *output layer O6*. Gambaran susunan arsitektur pada Tugas Akhir ini dapat dilihat pada **Gambar 4.1.**



Gambar 4.1 Arsitektur *Convolutional Neural Networks* untuk pengenalan wajah

Input model jaringan berupa hasil ekstraksi pola fitur ELBP berukuran 46 x 46 piksel dari *data preprocessing.*

Jenis lapisan jaringan pertama adalah jenis lapisan ekstraksi fitur. Hasil ekstraksi 46x46 pola fitur ditransformasi pada *Convolutional Layer (C1)* menjadi 20 *feature map 1 (FM1)* berukuran 42x42 pola fitur. *Feature map 1 (FM1)* akan melewati lapisan *Pooling Layer 2 (P2)* untuk mengurangi penggunaan parameter menjadi 20 *feature map 2 (FM2)* berukuran 14x14 pola fitur. Kemudian berlanjut ke lapisan berikutnya, *feature map 2 (FM2)* ditransformasikan pada *Convolutional Layer (C3)* menjadi 100 *feature map 3 (FM3)* berukuran 10x10 pola fitur. Sampai di lapisan ekstraksi fitur terakhir, *feature map 3 (FM3)* masuk ke dalam jenis lapisan untuk reduksi parameter yaitu *Pooling Layer 4 (P4)* menjadi 100 *feature map 4 (FM4)* berukuran 5x5 pola fitur.

Jenis lapisan jaringan selanjutnya adalah jenis lapisan *Fully Connected Layer*. *Feature map 4 (FM4)* dipandang sebagai neuron berjumlah 100x5x5 atau 2500 neuron. 2500 neuron terhubung penuh dengan 200 neuron pada *Hidden Layer 5 (H5)*. Hasil keluaran 200 neuron akan di masukkan ke jenis lapisan terakhir yaitu lapisan klasifikasi *Output Layer 6 (O6)* menghasilkan n keluaran, dengan n adalah jumlah kelas klassifikasi data. Metode klasifikasi yang digunakan pada Tugas Akhir ini adalah *Statistic Gradient Descent*.

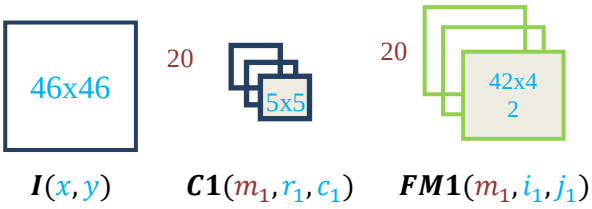
4.4 Pemodelan *Convolutional Neural Networks*

Konstruksi model jaringan mengacu pada arsitektur *Convolutional Neural Networks* yang sudah dibangun. Model berupa persamaan umum fungsi transfer antara pola fitur dengan pola fitur lainnya. Berikut penjabaran model pada setiap *layer* arsitektur jaringan.

4.6.1 Tahap Umpan Maju

Tahap umpan maju merupakan proses pembuatan dan pengecekan model.

a. *Convolutional Layer C1*



Gambar 4.2. Convolutional Layer C1

Convolutional Layer C1 merupakan layer ekstraksi pertama jaringan. Layer dikonstruksi dengan filter konvolusi, bertugas memfilter input gambar 46x46 dari hasil ekstraksi ELBP menjadi 20 *feature map* berukuran 42x42 yang dapat dilihat pada **Gambar 4.2**. Hasil susunan konstruksi dapat direpresentasikan sebagai *hyperparameter* dan *parameter* pada *layer* yang disajikan pada **Tabel 4.2** dan **Tabel 4.3**.

Tabel 4.2. *Hyperparameter Convolutional Layer C1*

Convolutional Layer C1		
No	Hyperparameter	Ukuran
1	*Depth*	20x42x42
2	*Stride*	1
3	*Zero-padding*	0

Tabel 4.3. Parameter *Convolutional Layer C1*

Convolutional Layer C1		
No	Parameter(*filter)*	Ukuran
1	Bobot	20x5x5
2	Bias	20

Fungsi aktivasi bipolar dipilih sebagai fungsi aktivasi pada *Convolutional Layer C1*. Keluaran fungsi aktivasi ini

memberikan batasan keluaran antara [-1,1]. Kelebihan fungsi ini menghasilkan nilai yang yang berpusat ke nol yang tidak terdapat pada fungsi sigmoid bipolar. Sehingga dirumuskan suatu fungsi umum sebagai berikut:

$$\boldsymbol{FM1}_{i_1,j_1}^{m_1} = \boldsymbol{f}\left(\boldsymbol{net}_{i_1,j_1}^{m_1}\right) = \boldsymbol{tanh}\left(\boldsymbol{b1}^{m_1} + \sum_{r_1=0}^{n_{r1}-1}\sum_{c_1=0}^{n_{r1}-1} \boldsymbol{C1}_{r_1,c_1}^{m_1} * \boldsymbol{I}_{(r_1+i_1),(c_1+j_1)}\right) \quad (4.1)$$

Selanjutnya didapatkan persamaan fungsi akhir berdasarkan *hyperparameter*, *parameter* dan fungsi aktivasi yang telah ditentukan pada proses konstruksi *Convolutional Layer C1* sebagai berikut :

$$\boldsymbol{FM1}_{(i_1,j_1)}^{(m_1)} = \boldsymbol{tanh}\left(\boldsymbol{b1}^{(m_1)} + \sum_{r_1=0}^{4}\sum_{c_1=0}^{4} \boldsymbol{C1}_{(r_1,c_1)}^{(m_1)} * \boldsymbol{I}_{((r_1+i_1),(c_1+j_1))}\right) \quad (4.2)$$

Keterangan variabel dan index pada *Convolutional Layer C1* adalah sebagai berikut:

$\boldsymbol{I}$: Input gambar
$\boldsymbol{b1}$: bias konvolusi layer ke-1
$\boldsymbol{C1}$: *filter* konvolusi layer ke-1
FM1 : *Feature Map-1*
x : index panjang baris input gambar
y : index panjang kolom input gambar
r_1 : index panjang baris *filter* konvolusi layer ke-1
c_1 : index panjang kolom *filter* konvolusi layer ke-1
i_1 : index baris *feature map* layer ke-1
j_1 : index kolom *feature map* layer ke-1
m_1 : index jumlah *feature pattern* ke-1
n_x : panjang baris input gambar
n_y : panjang kolom input gambar
n_{r_1} : panjang baris *filter* konvolusi layer ke-1
n_{c_1} : kolom *filter* konvolusi layer ke-1

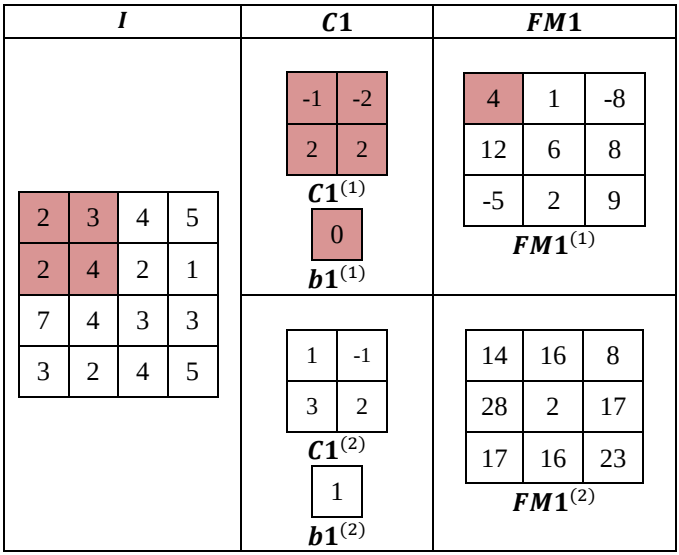
n_{i_1} : panjang baris *feature map* layer ke-1
n_{j_1} : panjang kolom *feature map* layer ke-1
n_{m_1} : jumlah jumlah *feature pattern* ke-1

Keterangan ukuran *Convolutional Layer C1* lebih terperinci disajikan pada **Tabel 4.4**.

Tabel 4.4. Elemen penyusun *Convolutional Layer C1*

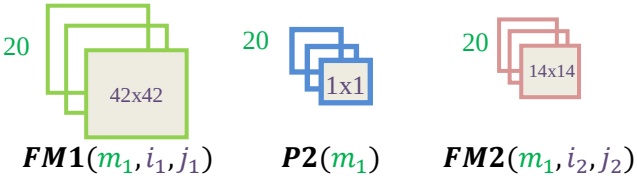
*Convolutional Layer C*1			
No	Nama	Variabel	Jum
1	Panjang baris input	n_x	46
2	Panjang kolom input	n_y	46
3	Panjang baris filter C1	n_{r1}	5
4	Panjang kolom filter C1	n_{c1}	5
5	Jumlah filter C1	n_{m1}	20
6	Panjang baris feature map 1	n_{i1}	42
7	Panjang kolom feature map 1	n_{j1}	42
8	Jumlah feature map 1	n_{m1}	20

Sebagai contoh, misal input *layer* berukuran 4 x 4 masuk kedalam *convolutional layer C1* dengan ukuran *filter C1* 2x2 berjumlah 2 tanpa fungsi aktivasi. Proses perhitungan ekstraksi disajikan pada **Gambar 4.3**.



Gambar 4.3. Contoh proses perhitungan ekstraksi pada *Convolutional Layer C1*

b. *Pooling Layer P2*



Gambar 4.4. *Pooling Layer* P2

Hasil ekstraksi fitur dari proses layer C1 diterima sebagai input pada *Pooling Layer P2*. Layer dikonstruksi dengan menerima input 20 *feature map* 1 dengan 20 *filter pooling* pergeseran 3 menghasilkan keluaran 20 *feature map 2* berukuran 14x14 yang dapat dlihat pada **Gambar 4.4**. Jenis *pooling layer*

yang dipilih berupa *pooling* yang dapat melakukan pembelajaran. Proses pembelajaran ini dengan cara menjumlahkan beberapa pengambilan bagian nilai input kemudian dikalikan dengan bobot pooling dan dijumlahkan dengan bias pooling. Jenis ini hampir sama dengan jenis *average pooling*, akan tetapi *average pooling* tidak memiliki proses pembelajaran dalam perhitungan samplingnya sehingga parameter tidak fleksibel. Dari hasil konstruksi ini dapat direpresentasikan sebagai *hyperparameter* dan *parameter* pada *layer* yang disajikan pada **Tabel 4.5** dan **Tabel 4.6**.

Tabel 4.5. *Hyperparameter Pooling Layer P2*

Pooling Layer P2		
No	Hyperparameter	Ukuran
1	*Depth*	20x14x14
2	*Stride*	3

Tabel 4.6. Parameter *Pooling Layer P2*

Pooling Layer P2		
No	Parameter (*filter)*	Jum
1	Bobot	20
2	Bias	20

Fungsi aktivasi bipolar dipilih sebagai fungsi aktivasi pada *Pooling Layer P2*. Sehingga persamaan fungsi akhir berdasarkan *hyperparameter*, *parameter* dan fungsi aktivasi yang telah ditentukan pada *Pooling Layer P2* adalah:

$$\boldsymbol{FM2}_{(i_2,j_2)}^{(m_1)} = \boldsymbol{f}\left(\boldsymbol{net}_{(i_2,j_2)}^{(m_1)}\right) = \boldsymbol{tanh}\left(\boldsymbol{b2}^{(m_1)} + \sum_{i_1=3i_2}^{3n_{i1}+2}\sum_{j_1=3j_2}^{3n_{j1}+2} \boldsymbol{P2}^{(m_1)} * \boldsymbol{FM1}_{(i_1,j_1)}^{(m_1)}\right) \quad (4.3)$$

Keterangan variabel dan index pada *Pooling Layer P2* adalah sebagai berikut:

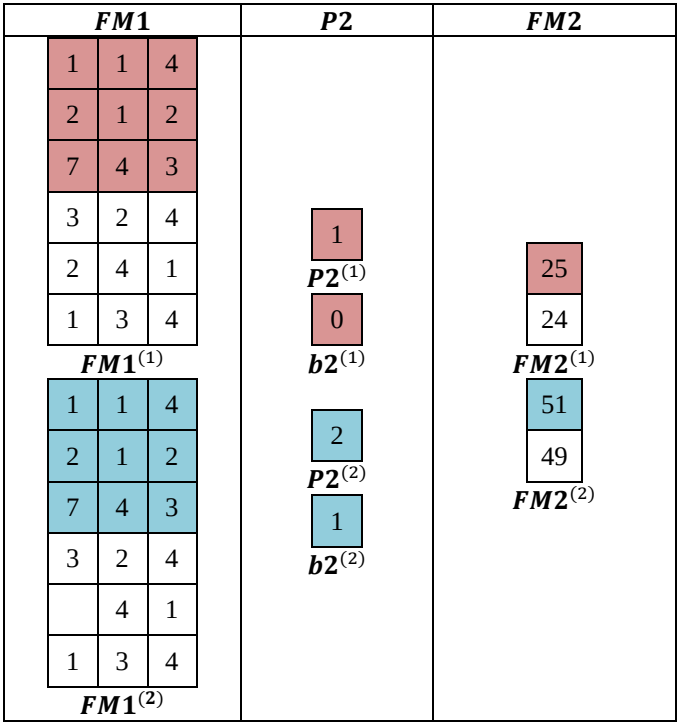
$\boldsymbol{b2}$: bias *filter* pooling layer ke-2
$\boldsymbol{P2}$: *filter* pooling layer ke-2
i_2 : index baris *feature map* ke-2
j_2 : index kolom *feature map* ke-2
n_{i_2} : panjang baris *feature map* ke-2
n_{j_2} : panjang kolom *feature map* ke-2

Keterangan ukuran *Pooling Layer P2* lebih terperinci disajikan pada **Tabel 4.7**.

Tabel 4.7. Elemen penyusun *Pooling Layer P2*

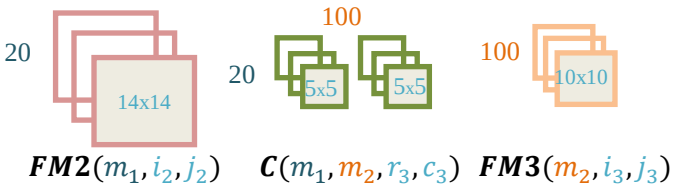
Pooling Layer P2			
No	Nama	Variabel	Jum
1	Jumlah filter pooling layer ke-2	n_{m1}	20
2	Jumlah filter bias pooling layer ke-2	n_{m1}	20
3	Panjang baris feature map 2	n_{i2}	14
4	Panjang kolom feature map 2	n_{j2}	14
5	Jumlah feature map 2	n_{m1}	20

Sebagai contoh, misal input berupa 2 *feature map 1* berukuran 6 x 3 masuk kedalam *pooling layer P2* berukuran 2 *filter P2* pergeseran(*stride)* 3 tanpa fungsi aktivasi. Proses perhitungan ekstraksi disajikan pada **Gambar 4.5**.



Gambar 4.5. Contoh proses perhitungan ekstraksi pada *Pooling Layer P2*

c. *Convolutional Layer C3*



Gambar 4.6. Convolutional Layer C3

Layer ini hampir sama dengan *Convolutional Layer C1*, perbedaan terletak pada jumlah *filter* yang lebih banyak. Lapisan dikonstruksi untuk menerima input 20 *feature map 2* berukuran 14x14 kemudian dilakukan transformasi menggunakan 20 *filter konvolusi* berukuran 5x5 berjumlah 100 hingga menghasilkan 100 *feature map* berukuran 10x10 yang dapat dilihat pada **Gambar 4.6**. Dari hasil konstruksi ini direpresentasikan sebagai *hyperparameter* dan *parameter* pada *layer* yang disajikan pada **Tabel 4.8** dan **Tabel 4.9**.

Tabel 4.8. *Hyperparameter Convolutional Layer C3*

Convolutional Layer C3		
No	Hyperparameter	Ukuran
1	*Depth*	100x10x10
2	*Stride*	1
3	*Zero-padding*	0

Tabel 4.9. Parameter *Convolutional Layer C3*

Convolutional Layer C3		
No	Parameter(*filter)*	Ukuran
1	Bobot	100x20x5x5
2	Bias	100

Fungsi aktivasi bipolar dipilih sebagai fungsi aktivasi pada *Convolutional Layer C3*. Sehingga dirumuskna suatu fungsi umum sebagai berikut:

$$\begin{aligned} \boldsymbol{FM3}_{(i_3,j_3)}^{(m_2)} &= \boldsymbol{f}\left(\boldsymbol{net}_{(i_3,j_3)}^{(m_2)}\right) \\ &= \boldsymbol{tanh}\Big(\boldsymbol{b3}^{(m_2)} + \\ &\qquad \sum_{m_1=0}^{n_{m1}-1} \sum_{r_3=0}^{n_{r3}-1} \sum_{c_3=0}^{n_{r3}-1} \boldsymbol{C3}_{(r_3,c_3)}^{(m_1,m_2)} * \boldsymbol{FM2}_{(r_3+i_3,c_3+j_3)}^{(m_1)}\Big) \end{aligned} \tag{4.4}$$

Selanjutnya didapatkan persamaan fungsi akhir berdasarkan *hyperparameter*, *parameter* dan fungsi aktivasi yang telah ditentukan pada *Convolutional Layer C3* sebagai berikut :

$$\boldsymbol{FM3}_{(i_3,j_3)}^{(m_2)} = \boldsymbol{tanh}\left(\sum_{m_1=0}^{19}\sum_{r_3=0}^{4}\sum_{c_3=0}^{4} \boldsymbol{C3}_{(r_3,c_3)}^{(m_1,m_2)} * \boldsymbol{FM2}_{(r_3+i_3,c_3+j_3)}^{(m_1)} + \boldsymbol{b3}^{(m_1,m_2)}\right) \quad (4.4)$$

Keterangan variabel dan index pada *Convolutional Layer C3* adalah sebagai berikut:

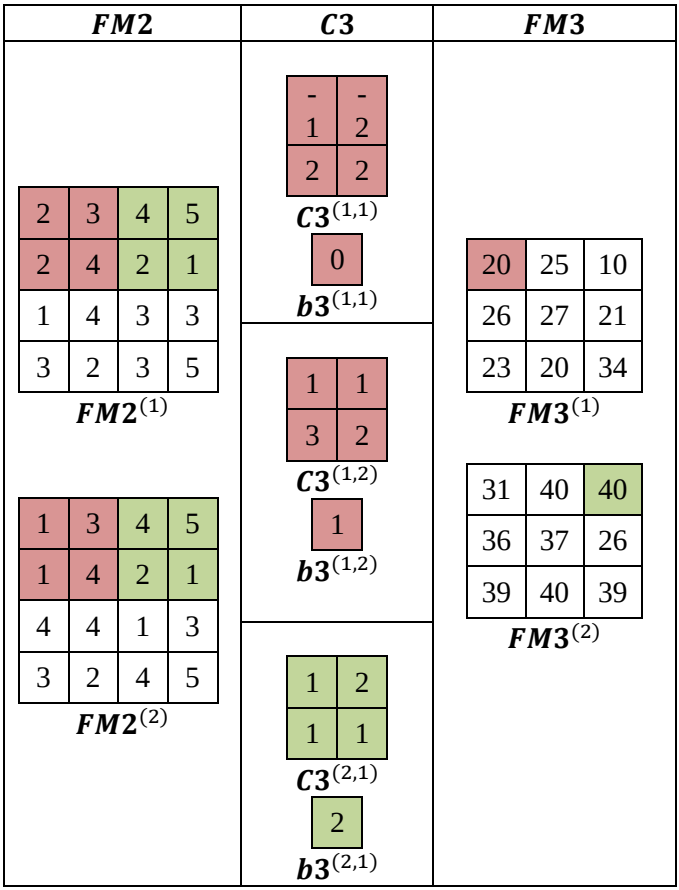
$\boldsymbol{b3}$: bias konvolusi layer ke-3
$\boldsymbol{C3}$: *filter* konvolusi layer ke-3
FM3 : Feature Map-3
m_2 : index kedalaman *feature pattern* ke-2
r_3 : index panjang baris *filter* konvolusi layer ke-3
c_3 : index panjang kolom *filter* konvolusi layer ke-3
i_3 : index baris *feature map* layer ke-3
j_3 : index kolom *feature map* layer ke-3
n_{m2} : jumlah kedalaman *feature pattern* ke-2
n_{r_3} : panjang baris *filter* konvolusi layer ke-3
n_{c_3} : kolom *filter* konvolusi layer ke-3
n_{i_3} : panjang baris *feature map* layer ke-3
n_{j_3} : panjang kolom *feature map* layer ke-3

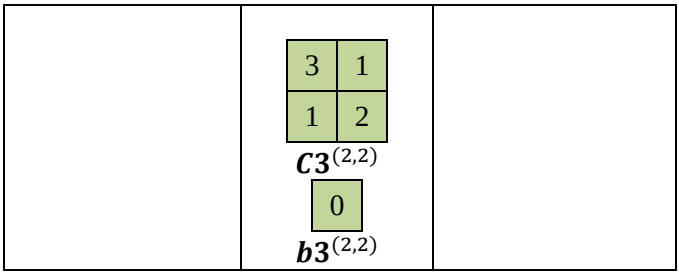
Keterangan ukuran *Convolutional Layer C3* lebih terperinci disajikan pada **Tabel 4.10**.

Tabel 4.10. Elemen penyusun *Convolutional Layer C3*

Convolutional Layer C3			
No	Nama	Variabel	Jum
1	Panjang baris filter C3	n_{r1}	5
2	Panjang kolom filter C3	n_{c1}	5
3	Jumlah filter C1	$n_{m1}xn_{m2}$	20x100
4	Jumlah bias filter C3	n_{m2}	100
7	Panjang baris feature map 3	n_{i1}	10
8	Panjang kolom feature map 3	n_{j2}	10
9	Jumlah feature map 3	n_{m2}	100
10	Kedalaman feture map 3	n_l	600

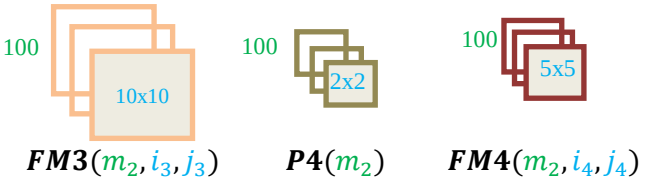
Sebagai contoh, misal hasil input 2 berukuran 4 x 4 berjumlah 2 masuk kedalam *convolutional layer C3* dengan ukuran *filter C1* 2x2 berjumlah 2 tanpa fungsi aktivasi. Proses perhitungan ekstraksi disajikan pada **Gambar 4.7**.





Gambar 4.7. Contoh proses perhitungan ekstraksi pada *Convolutional Layer C3*

d. *Pooling Layer P4*



Gambar 4.8. *Pooling Layer* P4

Hasil ekstraksi fitur dari proses *Convolutional Layer C3* diterima sebagai input pada *Pooling Layer P4*. Layer dikonstruksi dengan menerima input 20 *feature map 3* dengan 100 *filter pooling* pergeseran 2 menghasilkan keluaran 100 *feature map 4* berukuran 5x5 yang dapat dilihat pada **Gambar 4.8**. Jenis *pooling layer* yang dipilih sama dengan *Pooling Layer P2*. Dari hasil konstruksi ini dapat direpresentasikan sebagai *hyperparameter* dan *parameter* pada *layer* yang disajikan pada **Tabel 4.11** dan **Tabel 4.12**.

Tabel 4.11. *Hyperparameter Pooling Layer P4*

Pooling Layer P4		
No	Hyperparameter	Ukuran
1	*Depth*	100x5x5
2	*Stride*	2

Tabel 4.12. Parameter *Pooling Layer P4*

Pooling Layer P4		
No	Parameter (*filter)*	Jum
1	Bobot	100
2	Bias	100

Fungsi aktivasi bipolar dipilih sebagai fungsi aktivasi pada *Pooling Layer P4*. Sehingga persamaan fungsi akhir berdasarkan *hyperparameter*, *parameter* dan fungsi aktivasi yang telah ditentukan pada *Pooling Layer P2* adalah:

$$\boldsymbol{FM4}_{(i_4,j_4)}^{(m_2)} = \boldsymbol{f}\left(\boldsymbol{net}_{(i_4,j_4)}^{(m_2)}\right) = \boldsymbol{tanh}\left(\boldsymbol{b4}^{(m_2)} + \sum_{i_3=2i_4}^{3n_{i3}+2}\sum_{j_3=2j_4}^{3n_{j3}+2} \boldsymbol{P4}^{(m_2)} * \boldsymbol{FM3}_{(i_3,j_3)}^{(m_2)}\right) \quad (4.5)$$

Keterangan variabel dan index pada *Pooling Layer P4* adalah sebagai berikut:

$\boldsymbol{b4}$: bias *filter* pooling layer ke-4
$\boldsymbol{P4}$: *filter* pooling layer ke-4
i_4 : index baris *feature map* ke-4
j_4 : index kolom *feature map* ke-4
n_{i_4} : panjang baris *feature map* ke-4
n_{j_4} : panjang kolom *feature map* ke-4

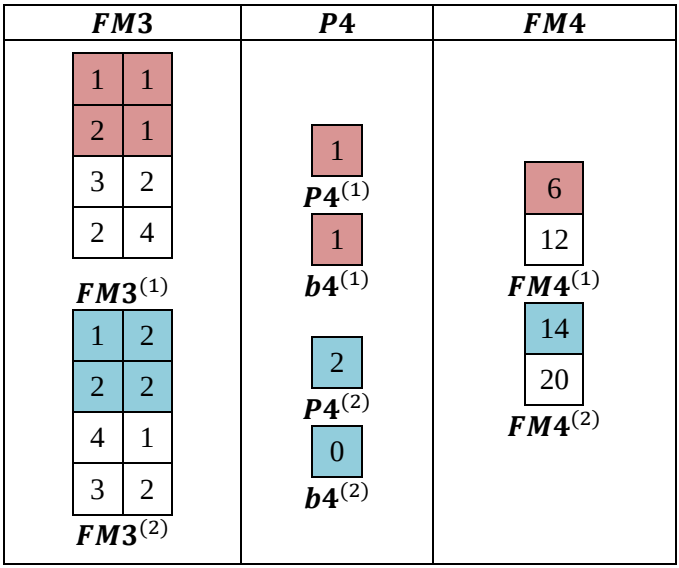
Keterangan ukuran *Pooling Layer P4* lebih terperinci disajikan pada **Tabel 4.13**.

Tabel 4.13. Elemen penyusun *Pooling Layer P4*

*Pooling Layer P*4			
No	Nama	Variabel	Jum
1	Jumlah filter pooling layer ke-4	n_{m2}	100
2	Jumlah filter bias pooling layer ke-4	n_{m2}	100
3	Panjang baris feature map 4	n_{i4}	5

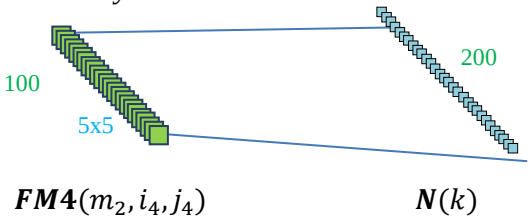
4	Panjang kolom feature map 4	n_{j4}	5
5	Jumlah feature map 4	n_{m2}	100

Sebagai contoh, misal input berupa 2 *feature map 3* berukuran 4 x2 masuk kedalam *pooling layer P4* dengan ukuran 2 *filter P4* pergeseran(*stride)* 2 tanpa fungsi aktivasi. Proses perhitungan ekstraksi disajikan pada **Gambar 4.9**.



Gambar 4.9. Contoh proses perhitungan ekstraksi pada *Pooling Layer P4*

e. *Hidden Layer H5*



Gambar 4.10. Jaringan *fully-connected layer*

Hidden layer diterapkan *Fully-Connected Layer* yaitu *layer* dikonstruksi dengan menghubungkan seluruh hasil akhir ekstraksi yaitu *feature map 4* dengan 200 neuron pada *hidden layer* yang terletak diujung arsitektur.

$$\boldsymbol{N^{(k)} = f\left(net^{(k)}\right)}$$
$$= \boldsymbol{tanh}\left(\boldsymbol{b5}^{(k)} + \sum_{m2=0}^{n_{m2}-1} \sum_{i_4=0}^{n_{i4}-1} \sum_{j_4=0}^{n_{j4}-1} \boldsymbol{H}_{(i,j)}^{(k,m_2)} * \boldsymbol{FM4}_{(i_4,j_4)}^{(m_2)} \right) \quad (4.6)$$

Keterangan variabel dan index pada *hidden layer* H5 adalah sebagai berikut:

$\boldsymbol{N}$: Neuron *hidden layer*
$\boldsymbol{b5}$: bias *hidden layer*
$\boldsymbol{H}$: bobot *hidden layer*
k : index jumlah neuron *hidden layer*
n_k : jumlah neuron *hidden layer*

f. *Output Layer O6*

Setiap unit yang ada di *hidden layer H5* terhubung dengan *output layer O6.* Jumlah neuron *output layer O6* disesuaikan dengan jumlah data(kelas) yang diklasifikasikan. Keluaran perhitungan berupa keluaran sebuah vektor aktual. Selanjutnya untuk mengetahui kesesuaian keluaran jaringan dilakukan perbandingan antara keluaran vektor aktual dengan keluaran vektor target. Keluaran jaringan dikatakan sesuai apabila keluaran aktual sudah hampir sama dengan keluaran target.

$$\boldsymbol{O^{(l)} = f\left(net^{(l)}\right)}$$
$$= \boldsymbol{tanh}\left(\boldsymbol{b6}^{(l)} + \sum_{k=0}^{n_k-1} \boldsymbol{W}_{(k)}^{(l)} * \boldsymbol{N5}_{(k)} \right) \quad (4.7)$$

Keterangan variabel dan index pada *neuron layer* O6 adalah sebagai berikut:

$\boldsymbol{O}$: neuron *output layer*
$\boldsymbol{b6}$: bias *output layer*

$\boldsymbol{W}$: bobot *output layer*
l : index jumlah neuron *output layer*
n_l : jumlah neuron *output layer*

4.6.2 Perhitungan MSE

Mean Squared Error (MSE) digunakan untuk menghitung perbedaan antara estimator dengan yang diestimasi yaitu perbedaan antara vektor aktual dengan keluaran vektor target. Pada dasarnya MSE menerapkan teori ruang hasil kali dalam. Andaikan f adalah sebuah fungsi kontinu pada $C[a,b]$ yang hendak kita hampiri dengan sebuah fungsi g pada $C[a,b]$, dan andaikan $C[a,b]$ memiliki hasil kali dalam,

$$\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \int_a^b f(x)g(x)\,dx$$

Dari persamaan ini dapat diketahui bahwa:

$$\|\boldsymbol{f}-\boldsymbol{g}\|^2 = \langle \boldsymbol{f}-\boldsymbol{g}, \boldsymbol{f}-\boldsymbol{g} \rangle = \int_a^b [f(x)-g(x)]^2 dx = MSE$$

Sehingga meminimalkan kesalahan kuadrat rata-rata sama artinya dengan meminimalkan $\|\boldsymbol{f}-\boldsymbol{g}\|^{\boldsymbol{2}}$**.** Hal ini dapat diterapkan pada permasalahan meminimalkan kesalahan dalam tahap *training* jaringan.

$$\boldsymbol{E} = \frac{1}{n_l} \sum_{\boldsymbol{l=0}}^{\boldsymbol{n_l-1}} \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right)^2 \tag{4.8}$$

Bagian $\frac{1}{n_l}$ berfungsi untuk mendapatkan nilai rata-rata kesalahan. Keterangan variabel dan index diatas pada perhitungan MSE adalah sebagai berikut:

$\boldsymbol{E}$: *error* pada model
$\boldsymbol{t}$: elemen vektor target(bipolar)

4.6.3 Tahap Umpan Mundur

Tujuan akhir *training* pada jaringan adalah untuk mencari gradien pada setiap *filter* $\boldsymbol{w}$ (bobot dan bias) berkenaan dengan keluaran:

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}_{ij}}$$

Sehingga dapat dilakukan pembaharuan *filter* secara bertahap menggunakan metode *stochastic gradient descent*.

$$\boldsymbol{w}_{ij} = \boldsymbol{w}_{ij} + \Delta \boldsymbol{w}_{ij} = \boldsymbol{w}_{ij} - \boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}_{ij}} \tag{4.9}$$

dengan $\boldsymbol{\eta}$ adalah *learning rate*.

1. O*utput layer* O6

Setiap unit keluran $\boldsymbol{O^l}$ menerima pola target t^l lalu informasi kesalahan lapisan keluaran (δ_l) dihitung. δ_l dikirim ke lapisan bawahnya dan digunakan untuk menghitung besar koreksi *filter*($\Delta \boldsymbol{W}_k^l$ dan $\Delta \boldsymbol{b6}^l$) antara *hidden layer* H5 dengan *output layer* O6.

$$\begin{aligned}
\delta_l &= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net_l}} \\
&= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^l} \frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net_l}} \\
&= -\frac{\partial \frac{1}{2}\left(\boldsymbol{t^l} - \boldsymbol{O^l}\right)^2}{\partial \boldsymbol{O}^l} \frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net_l}} \\
&= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \boldsymbol{f}'(\boldsymbol{net_l}) \\
&= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \boldsymbol{f}(\boldsymbol{net_l})(\boldsymbol{1} - \boldsymbol{f}(\boldsymbol{net_l}))
\end{aligned}$$

$$\begin{aligned}
\Delta \boldsymbol{W}_k^l &= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_k^l} \\
&= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net_l}} \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{W}_k^l}
\end{aligned}$$

$$= -\eta(-\delta_l)\frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{W}_k^l}$$
$$= \eta\left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \boldsymbol{f}(\boldsymbol{net_l})(\boldsymbol{1} - \boldsymbol{f}(\boldsymbol{net_l}))\boldsymbol{N5}_k \quad (4.10)$$

Oleh karena *neuron* bias = 1 maka $\frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{b6}^l} = 1$ sehingga,

$$\Delta \boldsymbol{b6}^l = -\eta \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b6}^l}$$
$$= -\eta \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net_l}} \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{b6}^l}$$
$$= -\eta(-\delta_l)$$
$$= \eta\delta_l \quad (4.11)$$

2. *Hidden Layer F5*

Pada setiap unit di *hidden layer* H5 dilakukan perhitungan informasi kesalahan *hidden layer* (δ^k) . δ^k kemudian digunakan untuk mengitung besar koreksi *filter*($\Delta \boldsymbol{H}_{i,j}^{k,m_2}$ dan $\Delta \boldsymbol{b5}^k$) antara *pooling layer P4* dengan *hidden layer* H5.

$$\delta^k = -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net^k}}$$
$$= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O^l}} \frac{\partial \boldsymbol{O^l}}{\boldsymbol{net^k}}$$
$$= -\frac{\partial \frac{1}{2}\left(\boldsymbol{t^l} - \boldsymbol{O^l}\right)^2}{\partial \boldsymbol{O^l}} \frac{\partial \boldsymbol{O^l}}{\partial \boldsymbol{net^k}}$$
$$= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \frac{\partial \boldsymbol{O^l}}{\partial \boldsymbol{N^k}} \frac{\partial \boldsymbol{N^k}}{\partial \boldsymbol{net^k}}$$
$$= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \frac{\partial \boldsymbol{O^l}}{\partial \boldsymbol{net_l}} \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N^k}} \boldsymbol{f'}(\boldsymbol{net^k})$$
$$= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \boldsymbol{f'}(\boldsymbol{net_l})\, \boldsymbol{W}_k^l \boldsymbol{f'}(\boldsymbol{net^k})$$
$$= \delta_l\, \boldsymbol{W}_k^l \boldsymbol{f}(\boldsymbol{net^k})(\boldsymbol{1} - \boldsymbol{f}\left(\boldsymbol{net^k}\right))$$

$$\Delta \boldsymbol{H}_{i,j}^{k,m_2} = -\eta \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{H}_{i,j}^{k,m_2}}$$

$$= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net^k}}\frac{\partial \boldsymbol{net^k}}{\partial \boldsymbol{H}_{i,j}^{k,m_2}}$$
$$= -\boldsymbol{\eta}(-\delta^k)\boldsymbol{FM4}_{i_4,j_4}^{m_2}$$
$$= \boldsymbol{\eta}\,\delta^k\boldsymbol{FM4}_{i_4,j_4}^{m_2} \quad (4.12)$$

Oleh karena *neuron* bias = 1 maka $\frac{\partial \boldsymbol{net^k}}{\partial \boldsymbol{b5}^k} = 1$ sehingga,

$$\Delta\boldsymbol{b5}^k = -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b5}^k}$$
$$= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net^k}}\frac{\partial \boldsymbol{net^k}}{\partial \boldsymbol{b5}^k}$$
$$= -\boldsymbol{\eta}(-\delta^k)$$
$$= \boldsymbol{\eta}\delta^k \quad (4.13)$$

3. *Pooling Layer P4*

Pada setiap unit di *pooling layer P4* dilakukan perhitungan informasi kesalahan *pooling layer P4* ($\delta_{i_4,j_4}^{m_2}$) . $\delta_{i_4,j_4}^{m_2}$ kemudian digunakan untuk mengitung besar koreksi *filter*($\Delta\boldsymbol{P4}^{m_2}$ dan $\Delta\boldsymbol{b4}^{m_2}$) antara *Convolutional Layer C3* dengan *pooling layer P4*.

$$\delta_{i_4,j_4}^{m_2} = -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}$$
$$= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^l}\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}$$
$$= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^l}\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}$$
$$= \left(\boldsymbol{t}^l - \boldsymbol{O}^l\right)\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}\frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}$$
$$= \left(\boldsymbol{t}^l - \boldsymbol{O}^l\right)\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net_l}}\frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{net^k}}\frac{\partial \boldsymbol{net^k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}\boldsymbol{f}'(\boldsymbol{net}_{i_4,j_4}^{m_2})$$
$$= \delta^k\boldsymbol{H}_{i,j}^{k,m_2}\boldsymbol{f}(\boldsymbol{net}_{i_4,j_4}^{m_2})(\boldsymbol{1} - \boldsymbol{f}\left(\boldsymbol{net}_{i_4,j_4}^{m_2}\right))$$

$$\begin{aligned}\Delta \boldsymbol{P4}^{m_2} &= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{P4}^{m_2}} \\ &= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}\frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{P4}^{m_2}} \\ &= -\boldsymbol{\eta}(-\delta_{i_4,j_4}^{m_2})\frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{P4}^{m_2}} \\ &= \boldsymbol{\eta}\delta_{i_4,j_4}^{m_2}\boldsymbol{f}(\boldsymbol{net}_{i_4,j_4}^{m_2})(\boldsymbol{1}-\boldsymbol{f}(\boldsymbol{net}_{i_4,j_4}^{m_2}))\boldsymbol{FM3}_{i_3,j_3}^{m_2} \quad (4.14)\end{aligned}$$

Oleh karena *neuron* bias = 1 maka $\frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{b4}^{m_2}} = 1$ sehingga,

$$\begin{aligned}\Delta \boldsymbol{b4}^{m_2} &= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b4}^{m_2}} \\ &= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}\frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{b4}^{m_2}} \\ &= -\boldsymbol{\eta}(-\delta_{i_4,j_4}^{m_2}) \quad (4.15) \\ &= \boldsymbol{\eta}\delta_{i_4,j_4}^{m_2}\end{aligned}$$

4. *Convolutional Layer C3*

Pada setiap unit di *Convolutional Layer C3* dilakukan perhitungan informasi kesalahan *Convolutional Layer C3* ($\delta_{i_3,j_3}^{m_2}$) . $\delta_{i_3,j_3}^{m_2}$ kemudian digunakan untuk mengitung besar koreksi *filter*($\Delta \boldsymbol{C3}_{r_3,c_3}^{m_1,m_2}$ dan $\Delta \boldsymbol{b3}^{m_2}$) antara *Pooling Layer P2* dengan *Convolutional Layer C3*.

$$\begin{aligned}\delta_{i_3,j_3}^{m_2} &= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}} \\ &= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^{l}}\frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}} \\ &= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^{l}}\frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{N}^{k}}\frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}\end{aligned}$$

$$= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \frac{\partial \boldsymbol{O^l}}{\partial \boldsymbol{N^k}} \frac{\partial \boldsymbol{N^k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}$$

$$= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \frac{\partial \boldsymbol{O^l}}{\partial \boldsymbol{net_l}} \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N^k}} \frac{\partial \boldsymbol{N^k}}{\partial \boldsymbol{net^k}} \frac{\partial \boldsymbol{net^k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}$$

$$= \left(\boldsymbol{t^l} - \boldsymbol{O^l}\right) \frac{\partial \boldsymbol{O^l}}{\partial \boldsymbol{net_l}} \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N^k}} \frac{\partial \boldsymbol{N^k}}{\partial \boldsymbol{net^k}} \frac{\partial \boldsymbol{net^k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}$$

$$= \delta_{i_4,j_4}^{m_2} \boldsymbol{P4}^{m_2} \boldsymbol{f}(\boldsymbol{net}_{i_3,j_3}^{m_2})(\boldsymbol{1} - \boldsymbol{f}\left(\boldsymbol{net}_{i_3,j_3}^{m_2}\right))$$

$$\Delta \boldsymbol{C3}_{r_3,c}^{m_1,} = -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{C3}_{r_3,c_3}^{m_1,m_2}}$$

$$= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{C3}_{r_3,c_3}^{m_1,m_2}}$$

$$= -\boldsymbol{\eta}(-\delta_{i_3,j_3}^{m_2}) \frac{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{C3}_{r_3,c_3}^{m_1,m_2}}$$

$$= \boldsymbol{\eta} \delta_{i_3,j_3}^{m_2} \boldsymbol{f}(\boldsymbol{net}_{i_3,j_3}^{m_2}) (\boldsymbol{1} - \boldsymbol{f}(\boldsymbol{net}_{i_3,j_3}^{m_2})) \boldsymbol{FM2}_{r_3+i_3,c_3+j_3}^{m_1} \quad (4.16)$$

Oleh karena *neuron* bias = 1 maka $\frac{\partial \boldsymbol{net}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{b3}^{m_2}} = 1$ sehingga,

$$\Delta \boldsymbol{b3}^{m_2} = -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b3}^{m_2}}$$

$$= -\boldsymbol{\eta}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}^{m_2}_{i_3,j_3}}\frac{\partial \boldsymbol{net}^{m_2}_{i_3,j_3}}{\partial \boldsymbol{b3}^{m_2}}$$
$$= -\boldsymbol{\eta}(-\delta^{m_2}_{i_3,j_3})$$
$$= \boldsymbol{\eta}\delta^{m_2}_{i_3,j_3} \quad (4.17)$$

5. *Pooling Layer P2*

Pada setiap unit di *Pooling Layer P2* dilakukan perhitungan informasi kesalahan *Pooling Layer P2* ($\delta^{m_1}_{i_2,j_2}$) . $\delta^{m_1}_{i_2,j_2}$ kemudian digunakan untuk mengitung besar koreksi *filter*($\Delta\boldsymbol{P2}^{m_1}$ dan $\Delta\boldsymbol{b2}^{m_1}$)antara *Convolutional Layer C1* dengan *Pooling Layer P2*.

$$\delta^{m_1}_{i_2,j_2} = -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}^{m_1}_{i_2,j_2}}$$
$$= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^l}\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net}^{m_1}_{i_2,j_2}}$$
$$= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^l}\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{net}^{m_1}_{i_2,j_2}}$$
$$= \left(\boldsymbol{t}^l - \boldsymbol{O}^l\right)\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}\frac{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}{\partial \boldsymbol{net}^{m_1}_{i_2,j_2}}$$
$$= \left(\boldsymbol{t}^l - \boldsymbol{O}^l\right)\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net_l}}\frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{net}^k}\frac{\partial \boldsymbol{net}^k}{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}$$
$$\frac{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}{\partial \boldsymbol{FM3}^{m_2}_{i_3,j_3}}\frac{\partial \boldsymbol{FM3}^{m_2}_{i_3,j_3}}{\partial \boldsymbol{net}^{m_1}_{i_2,j_2}}$$
$$= \left(\boldsymbol{t}^l - \boldsymbol{O}^l\right)\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net_l}}\frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{net}^k}\frac{\partial \boldsymbol{net}^k}{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}$$
$$\frac{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}{\partial \boldsymbol{net}^{m_2}_{i_4,j_4}}\frac{\partial \boldsymbol{net}^{m_2}_{i_4,j_4}}{\partial \boldsymbol{FM3}^{m_2}_{i_3,j_3}}\frac{\partial \boldsymbol{FM3}^{m_2}_{i_3,j_3}}{\boldsymbol{net}^{m_1}_{i_2,j_2}}$$
$$= \left(\boldsymbol{t}^l - \boldsymbol{O}^l\right)\frac{\partial \boldsymbol{O}^l}{\partial \boldsymbol{net_l}}\frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N}^k}\frac{\partial \boldsymbol{N}^k}{\partial \boldsymbol{net}^k}\frac{\partial \boldsymbol{net}^k}{\partial \boldsymbol{FM4}^{m_2}_{i_4,j_4}}$$

$$\frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}}{\boldsymbol{net}_{i_2,j_2}^{m_1}}$$

$$= \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{l}} \frac{\partial \boldsymbol{net}_{l}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}^{k}} \frac{\partial \boldsymbol{net}^{k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}$$

$$\frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\boldsymbol{net}_{i_3,j_3}^{m_2}} \frac{\boldsymbol{net}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}}$$

$$\frac{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}}{\boldsymbol{net}_{i_2,j_2}^{m_1}}$$

$$= \delta_{i_4,j_4}^{m_2} \boldsymbol{P2}^{m_1} \boldsymbol{f}(\boldsymbol{net}_{i_2,j_2}^{m_1})(\boldsymbol{1} - \boldsymbol{f}\left(\boldsymbol{net}_{i_2,j_2}^{m_1}\right))$$

$$\Delta \boldsymbol{P2}^{m_1} = -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{P2}^{m_1}}$$

$$= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{P2}^{m_1}}$$

$$= -\boldsymbol{\eta}(-\delta_{i_2,j_2}^{m_1}) \frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{P2}^{m_1}}$$

$$= \boldsymbol{\eta} \delta_{i_2,j_2}^{m_1} \boldsymbol{f}(\boldsymbol{net}_{i_2,j_2}^{m_1})(\boldsymbol{1} - \boldsymbol{f}(\boldsymbol{net}_{i_2,j_2}^{m_1})) \boldsymbol{FM1}_{i_1,j_1}^{m_1} \quad (4.18)$$

Oleh karena *neuron* bias = 1 maka $\frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{b2}^{m_1}} = 1$ sehingga,

$$\Delta \boldsymbol{b2}^{m_1} = -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b2}^{m_1}}$$

$$= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{b2}^{m_1}}$$

$$= -\boldsymbol{\eta}(-\delta_{i_2,j_2}^{m_1})$$

$$= \boldsymbol{\eta} \delta_{i_2,j_2}^{m_1} \quad (4.19)$$

6. *Convolutional Layer C1*

Pada setiap unit di *Convolutional Layer C1* dilakukan perhitungan informasi kesalahan *Convolutional Layer C1* ($\delta_{i_1,j_1}^{m_1}$).

$\delta_{i_1,j_1}^{m_1}$ kemudian digunakan untuk mengitung besar koreksi *filter*($\Delta \boldsymbol{C1}_{r_1,c_1}^{m_1}$ dan $\Delta \boldsymbol{b1}^{m_1}$) antara Input dengan *Convolutional Layer C1*.

$$
\begin{aligned}
\delta_{i_1,j_1}^{m_1} &= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^{l}} \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= -\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}^{l}} \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{\boldsymbol{l}}} \frac{\partial \boldsymbol{net}_{\boldsymbol{l}}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}^{\boldsymbol{k}}} \frac{\partial \boldsymbol{net}^{\boldsymbol{k}}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \\
&\quad \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{\boldsymbol{l}}} \frac{\partial \boldsymbol{net}_{\boldsymbol{l}}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}^{\boldsymbol{k}}} \frac{\partial \boldsymbol{net}^{\boldsymbol{k}}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \\
&\quad \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{\boldsymbol{l}}} \frac{\partial \boldsymbol{net}_{\boldsymbol{l}}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}^{\boldsymbol{k}}} \frac{\partial \boldsymbol{net}^{\boldsymbol{k}}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \\
&\quad \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}}{\boldsymbol{net}_{i_1,j_1}^{m_1}} \\
&= \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net}_{\boldsymbol{l}}} \frac{\partial \boldsymbol{net}_{\boldsymbol{l}}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}^{\boldsymbol{k}}} \frac{\partial \boldsymbol{net}^{\boldsymbol{k}}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \\
&\quad \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{FM1}_{i_1,j_1}^{m_1}}
\end{aligned}
$$

$$
\begin{aligned}
& \frac{\partial \boldsymbol{FM1}_{i_1,j_1}^{m_1}}{\boldsymbol{net}_{i_1,j_1}^{m_1}} \\
= \; & \left(\boldsymbol{t}^{l} - \boldsymbol{O}^{l}\right) \frac{\partial \boldsymbol{O}^{l}}{\partial \boldsymbol{net_l}} \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{N}^{k}} \frac{\partial \boldsymbol{N}^{k}}{\partial \boldsymbol{net}^{k}} \frac{\partial \boldsymbol{net}^{k}}{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}} \\
& \frac{\partial \boldsymbol{FM4}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}} \frac{\partial \boldsymbol{net}_{i_4,j_4}^{m_2}}{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}} \frac{\partial \boldsymbol{FM3}_{i_3,j_3}^{m_2}}{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{FM2}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}} \\
& \frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{FM1}_{i_1,j_1}^{m_1}} \frac{\partial \boldsymbol{FM1}_{i_1,j_1}^{m_1}}{\boldsymbol{net}_{i_1,j_1}^{m_1}} \\
= \; & \delta_{i_4,j_4}^{m_2} \boldsymbol{P2}^{m_1} \boldsymbol{f}(\boldsymbol{net}_{i_2,j_2}^{m_1}) \left(\mathbf{1} - \boldsymbol{f}\left(\boldsymbol{net}_{i_2,j_2}^{m_1}\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
\Delta \boldsymbol{C1}_{r_1,c}^{m_1} &= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{C1}_{r_1,c_1}^{m_1}} \\
&= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{C1}_{r_1,c_1}^{m_1}} \\
&= -\boldsymbol{\eta}(-\delta_{i_1,j_1}^{m_1}) \frac{\partial \boldsymbol{net_l}}{\partial \boldsymbol{C1}_{r_1,c_1}^{m_1}} \\
&= \boldsymbol{\eta} \delta_{i_1,j_1}^{m_1} \; \boldsymbol{f}(\boldsymbol{net}_{i_2,j_2}^{m_1}) \\
&\quad (\mathbf{1} - \boldsymbol{f}(\boldsymbol{net}_{i_2,j_2}^{m_1})) \boldsymbol{I}_{(r_1+i_1),(c_1+j_1)} \qquad (4.20)
\end{aligned}
$$

Oleh karena *neuron* bias = 1 maka $\frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{b1}^{m_1}} = 1$ sehingga,

$$
\begin{aligned}
\Delta \boldsymbol{b1}^{m_1} &= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b1}^{m_1}} \\
&= -\boldsymbol{\eta} \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}} \frac{\partial \boldsymbol{net}_{i_2,j_2}^{m_1}}{\partial \boldsymbol{b1}^{m_1}} \\
&= -\boldsymbol{\eta}(-\delta_{i_1,j_1}^{m_1}) \\
&= \boldsymbol{\eta} \delta_{i_1,j_1}^{m_1} \qquad (4.21)
\end{aligned}
$$

4.5 Konfigurasi parameter Convolutiona Neural Networks

Distribusi keluaran dari inisialisasi neuron secara acak mempunyai varians yang menyebar pada sejumlah input.

$$\begin{aligned}\text{Var(s)} &= Var\left(\sum_{i}^{n} w_i x_i\right) \\ &= \sum_{i}^{n} Var(w_i x_i) \\ &= \sum_{i}^{n} [E(w_i)]^2 Var(x_i) + E[(x_i)]^2 Var(w_i) \\ &\quad + Var(x_i)Var(w_i) \\ &= \sum_{i}^{n} Var(x_i)Var(w_i) \\ &= (nVar(w))Var(x)) \end{aligned} \tag{4.22}$$

Selanjutnya $nVar(w) = 1$

$$nVar(w) = Var\left(w/\sqrt{n}\right) = 1$$

Dengan demikian inisialisasi nilai *parameter* bobot dilakukan dengan membagi bilangan acak dari fungsi standar distribusi normal $N(0,1)$ (*mean=0* dan *variance*=1) dengan akar dari jumlah parameter bobot:

$$w(x) = f(r)\frac{1}{\sqrt{n}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r-\mu)^2}{2\sigma^2}} \frac{1}{\sqrt{n}} = \frac{1}{\sqrt{2\pi}} e^{-\frac{r^2}{2}} \frac{1}{\sqrt{n}} \tag{4.23}$$

r : bilangan acak *(random)*
n : jumlah parameter bobot
σ : standar deviasi
μ : *mean* atau ekspektasi

Sedangkan parameter bias diinisialisasi dengan angka nol.

BAB V
IMPLEMENTASI SISTEM

Pada bab ini, dibahas mengenai langkah-langkah dalam pengimplemetasian sistem berdasarkan desain sistem yang telah dirancang.

5.1 Lingkungan Hardware dan Software

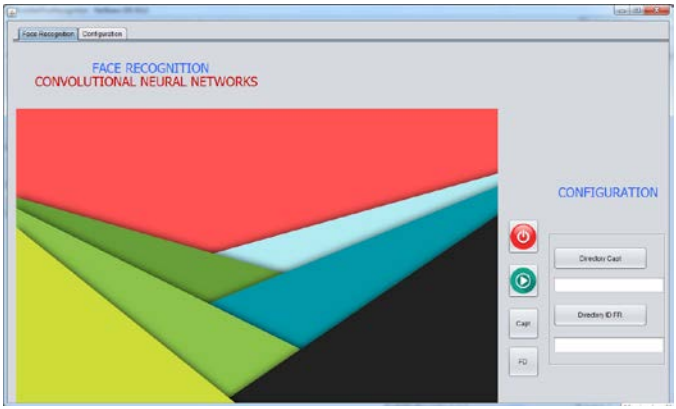
Lingkungan perancangan sistem dibangun dari dua lingkungan yaitu lingkungan *software* dan lingkungan *hardware*. Spesifikasi lingkungan perancangan sistem secara lengkap dapat dilihat pada **Tabel 5.1**.

Tabel 5.1. Spesifikasi *software* dan *hardware*

Lingkungan	Spesifikasi	
Hardware	Processor	Intel® Core™ i3 2.13 GHz
	RAM	3 GB
	WebCam	5MP 640x320 piksel
Software	Sistem Operasi	Windows 7 Ultimate 32 Bit
	Tools	Netbeans IDE 8.0.2
		MatLab 2010

5.2 Implementasi UI Face Recognition

Desain *User Iterface* dibuat seluas mungkin untuk kemudahan dalam perekeman menggunakan webcam secara *real-time* dapat dilihat pada **Gambar 5.1**.



Gambar 5.1. *User Interface Face Detection* dan *Face Recognition*

5.3 Implementasi WebCam dan deteksi wajah

1. Perekaman secara *Real-time*

Konfigurasi untuk menghubungkan komputer dengan webcam menggunakan bantuan *library* OpenCV dapat dilakukan dengan mendeklarasikan kelas VideoCapture dari *library* sebagai berikut:

```
//pada luar kelas
import org.opencv.videoio.VideoCapture;
//didalam kelas
VideoCapture webSource = new VideoCapture(0);
```

Perekaman kamera secara *real-time* diimplementasikan dengan kelas yang mengimplementasikan Runnable. Dalam Kelas implement Runnable akan ada `@override` fungsi run. Fungsi ini bekerja secara terus menerus atau digunakan sebagai tempat peletakan semua yang berhubungan dengan operasi pada kamera

nantinya. Didalam fungsi ini dipanggil fungsi penerima gambar tiap frame *real-time* dari webcam sebagai berikut:

```
webSource.retrieve(frame);
```

2. Deteksi multi wajah

OpenCV sudah menyediakan algoritma untuk deteksi multi wajah. Deteksi multi wajah menggunakan algoritma standar Paul Viola dan Michael Jones dalam jurnalnya "Rapid Object Detection using a Boosted Cascade of Simple Features" tahun 2001 yang sudah tersedia di dalam *library*. Selanjutnya tinggal mencari alamat file algoritma deteksi multi wajah pada folder OpenCV sebagai berikut:

```
CascadeClassifier faceDetector = new
CascadeClassifier("...\\opencv\\sources\\data\\h
aarcascades\\haarcascade_frontalface_alt.xml");
```

Pemanggilan fungsi deteksi multi wajah akan diberi *bounding box* yang didefinisikan pada fungsi run:

```
faceDetector.detectMultiScale(frame,
faceDetections);

Imgproc.rectangle(frame, new Point(rect.x,
rect.y), new Point(rect.x + rect.width, rect.y +
rect.height),
new Scalar(0, 255,0));
```

5.4 Implementasi Pengambilan Gambar Real-Time

Selanjutnya setiap wajah yang terdeteksi dari hasil *bounding box* diubah ukurannya menjadi 48x48 piksel dan dikonversikan ke dalam *grayscale*, disimpan menjadi sebuah file berekstensi PNG.

```
rect_Crop = new Rect(rect.x, rect.y, rect.width,
rect.height);
image_roi = new Mat(frame,rect_Crop);
Imgproc.resize(image_roi, newFrame, sz);
Imgproc.cvtColor(newFrame,newFrame,
Imgproc.COLOR_BGR2GRAY);
f=new     File     (     directorysavePhoto     +
"\\gambar"+gb_kei+".png");
Imgcodecs.imwrite(f.getPath(), newFrame);
```

5.5 Implementasi Preprocecssing data

Preprocessing data dilakukan dengan mengektraksi gambar grayscale 48x48 dengan menggunakan ELBP menjadi fitur berukuran 46x46:

```
for(int n=0; n<neighbors; n++) {
x = (float) ((radius)*
Math.cos(2.*Math.PI*n/(float)neighbors));
y = (float) ((radius)* -
Math.sin(2.*Math.PI*n/(float)neighbors));
// relative indices
fx = (int) Math.floor(x);
fy = (int) Math.floor(y);
cx = (int) Math.ceil(x);
cy = (int) Math.ceil(y);
// fractional part
ty = y - fy;
tx = x - fx;
// set interpolation weights
w1 = (1 - tx) * (1 - ty);
w2 =      tx  * (1 - ty);
w3 = (1 - tx) *      ty;
w4 =      tx  *      ty;
// iterate through your data
for(i=radius; i < src.length-radius;i++) {
for(j=radius;j < src[0].length-radius;j++) {
```

```
    t = w1*(src[i+fy][j+fx]&0x0ff) +
w2*(src[i+fy][j+cx]&0x0ff) +
w3*(src[i+cy][j+fx]&0x0ff) +
w4*(src[i+cy][j+cx]&0x0ff);

    if ((t > (src[i][j]&0x0ff)) && (Math.abs(t-
(src[i][j]&0x0ff)) > 0.1))
        dst[i-radius][j-radius] += n;}}}
```

Inisialisasi ELBP pada saat tahap training dan deteksi adalah *Neighborhoods* P*(pattern)* = 8 dan R*(radius)* = 1.0 diimplemetasikan pada fungsi run :

```
BismillahLBP.getExtendedLBP(BismillahImage.matTo
Byte(newFrame),1,15)
```

5.6 Implementasi Model CNN

5.6.1 Inisialisasi *Hyperparameter* dan Parameter Propagasi Maju

Berikut deklarasi variabel awal untuk arsitektur jaringan dengan kedalaman 7 *layer* propagasi maju dari rancangan desain sistem:

```
private static double bobotKonvolusiL1[][][] =
new double[20][5][5];
private static double biasKonvolusiL1[] = new
double[20];

//Feature Map1
private static final byte tinggifeatureMap1 =
42;
private static final byte lebarfeatureMap1 = 42;
private static double featureMap1[][][] = new
double[20][42][42];

private static double bobotPoolingL2[] = new
double[20];
```

```
private static double biasPoolingL2[] = new
double[20];

//Hyperparameter Layer 2 (Pooling)
private static final byte
tinggiFilterMaxPoolingL2 = 3;
private static final byte
lebarFilterMaxPoolingL2 = 3;

//Feature Map2
private static double featureMap2[][][] = new
double[20][14][14];
private static final byte tinggifeatureMap2 =
14;
private static final byte lebarfeatureMap2 = 14;

private static double bobotKonvolusiL3[][][][] =
new double[100][20][5][5];
private static double biasKonvolusiL3[][][] =
new double[100][10][10];

//Feature Map3
private static double featureMap3[][][] = new
double[100][10][10];
private static final byte tinggifeatureMap3 =
10;
private static final byte lebarfeatureMap3 = 10;

private static double bobotPoolingL4[] = new
double[100];
private static double biasPoolingL4[] = new
double[100];

//Hyperparameter Layer 4 (MaxPooling)
private static final byte
tinggiFilterMaxPoolingL4 = 2;
private static final byte
lebarFilterMaxPoolingL4 = 2;

//Feature Map4
```

```
private static double featureMap4[][][] = new
double[100][5][5];
private static final byte tinggifeatureMap4 = 5;
private static final byte lebarfeatureMap4 = 5;

private static double bobotXtoZ [][][][] = new
double [200][100][5][5];
private static double biasXtoZ [] = new
double[200]; //bias = neuron z =200
private static double z_net[] =  new
double[200];

private static double bobotZtoY [][];
private static double biasZtoY[];
private static double y_net[];
private static double target[][];
```

5.6.2 Inisialisasi *Hyperparameter* dan Parameter Propagasi Mundur

Berikut deklarasi variabel awal untuk arsitektur jaringan dengan kedalaman 7 *layer* propagasi mundur dari rancangan desain sistem:

```
private static double deltaY[];
private static double deltaBobotZtoY [][];
private static double deltabiasZtoY[];

private static double deltaZ[] = new
double[200];
private static double deltaBobotXtoZ [][][][] =
new double [200][100][5][5];
private static double deltaBiasXtoZ [] = new
double[200]; //bias = neuron z =200

private static double deltaFeatureMap4[][][] =
new double[100][5][5];
private static double deltaBobotPoolingL4[] =
new double[100];
```

```
private static double deltaBiasPoolingL4[] = new
double[100];

private static double deltaFeatureMap3[][][] =
new double[100][10][10];
private static double
deltaBobotKonvolusiL3[][][][] = new
double[100][20][5][5];
private static double deltaBiasKonvolusiL3[][][]
= new double[100][10][10];

private static double deltaBobotPoolingL2[] =
new double[20];
private static double deltaBiasPoolingL2[] = new
double[20];
private static double deltaFeatureMap2[][][] =
new double[20][14][14];

private static double
deltaBobotKonvolusiL1[][][] = new
double[20][5][5];
private static double deltaBiasKonvolusiL1[] =
new double[20];
private static double deltaFeatureMap1[][][] =
new double[20][42][42];
```

5.6.3 Implementasi Convolutional Layer C1

1. Propagasi Maju

```
//Layer 1 Konvolusi-1
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_1
for(jLoop = 0;jLoop < 42;jLoop++){ //
tinggi_feature_Map_1 = 46 - (5-1) = 42
for(kLoop = 0;kLoop < 42;kLoop++){ //
lebar_feature_Map_1 = 46 - (5-1) = 42
//\\Konvolusi 1//\\
featureMap1[iLoop][jLoop][kLoop] = 0;
```

```
for(iKern = jLoop;iKern < (jLoop+5);iKern++){
//tinggi_kernel_konvolusi_1 = 5
for(jKern = kLoop;jKern < (kLoop+5);jKern++){
//lebar_kernel_konvolusi_1 = 5
featureMap1[iLoop][jLoop][kLoop] +=
(image[iKern][jKern]&
0xff)*bobotKonvolusiL1[iLoop][iKern-
jLoop][jKern-kLoop];
}
}
//\\Konvolusi 1//\\
featureMap1[iLoop][jLoop][kLoop] =
Math.tanh(featureMap1[iLoop][jLoop][kLoop] +
biasKonvolusiL1[iLoop]);
}
}
}
```

2. Propagasi Mundur

```
//delta di unit featureMap1(X) (bias tidak
diikutsertakan dalam perhitungan, karena bias =
1)
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_1
for(jLoop = 0;jLoop < 42;jLoop++){ //
tinggi_feature_Map_1 = 46 - (5-1) = 42
for(kLoop = 0;kLoop < 42;kLoop++){ //
lebar_feature_Map_1 = 46 - (5-1) = 42
deltaFeatureMap1[iLoop][jLoop][kLoop] =
bobotPoolingL2[iLoop] *
deltaFeatureMap2[iLoop][jLoop/3][kLoop/3]*(1-
Math.tanh(featureMap1[iLoop][jLoop][kLoop])*Math
.tanh(featureMap1[iLoop][jLoop][kLoop]));}}}

//delta di bobot L1 fMap1 ke fMap2
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_1
```

```
for(jLoop = 0;jLoop < 5;jLoop++){ //
tinggi_feature_Map_1 = 46 - (5-1) = 42
for(kLoop = 0;kLoop < 5;kLoop++){ //
lebar_feature_Map_1 = 46 - (5-1) = 42
deltaBobotKonvolusiL1[iLoop][jLoop][kLoop] = 0;
for(iKern = 0;iKern < 42;iKern++){
//tinggi_kernel_konvolusi_1 = 5
for(jKern = 0;jKern < 42;jKern++){
//lebar_kernel_konvolusi_1 = 5
deltaBobotKonvolusiL1[iLoop][jLoop][kLoop] +=
deltaFeatureMap1[iLoop][iKern][jKern]*(image[NLo
op1][iKern+jLoop][jKern+kLoop]& 0xff);}}
deltaBobotKonvolusiL1[iLoop][jLoop][kLoop] *=
alpha;
}
}
}

for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_1
deltaBiasKonvolusiL1[iLoop] = 0;
for(iKern = 0;iKern < 42;iKern++){
//tinggi_kernel_konvolusi_1 = 5
for(jKern = 0;jKern < 42;jKern++){
//lebar_kernel_konvolusi_1 = 5
deltaBiasKonvolusiL1[iLoop] +=
deltaFeatureMap1[iLoop][iKern][jKern];}}deltaBia
sKonvolusiL1[iLoop] *=alpha;
}
```

5.6.4 Implementasi *Pooling Layer* P2

1. Propagasi Maju

```
//Layer 2 Pooling-1
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_2
for(jLoop = 0;jLoop < 14;jLoop++){ //
tinggi_feture Map_2 = 42 /
tinggi_kernel_pooling_1 = 42 / 3 = 14
```

```
for(kLoop = 0;kLoop < 14;kLoop++){ //
lebar_feture Map_2 = 42 / lebar_kernel_pooling_1
= 42 / 3 = 14
featureMap2[iLoop][jLoop][kLoop] = 0;
//\\Pooling//\\
for(iKern = (byte) (jLoop*3);iKern <
((jLoop+1)*3);iKern++){ //
tinggi_kernel_pooling_1 = 3
for(jKern = (byte) (kLoop*3);jKern <
((kLoop+1)*3);jKern++){ //
lebar_kernel_pooling_1 = 3
featureMap2[iLoop][jLoop][kLoop]
+=featureMap1[iLoop][iKern][jKern];}}
featureMap2[iLoop][jLoop][kLoop] =
Math.tanh(bobotPoolingL2[iLoop]*featureMap2[iLoo
p][jLoop][kLoop]+biasPoolingL2[iLoop]);
//\\Pooling//\\}}}
```

2. Propagasi Mundur

```
//delta di unit featureMap2(X) (bias tidak
diikutsertakan dalam perhitungan, karena bias =
1)
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_2
for(jLoop = 0;jLoop < 100;jLoop++){ // 100
feature_Map_3

for(byte row = 0;row<14;row++){
if(row<5){ //n=5
for(byte iRow = row,iRow2 = 0;iRow>=0;iRow--
,iRow2++){
for(byte colm = 0;colm<14;colm++){

if(colm<5){ //n=5
for(byte iColm = colm,iColm2 =0;iColm>=0;iColm--
,iColm2++){
```

```
//
System.out.println("\t1["+iRow+"]["+iColm+"]"+"\
t["+iRow2+"]["+iColm2+"]");
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.println("\t2["+iRow+"]["+iColm+"]"+"\
t["+iRow2+"]["+iColm2+"]");
}
}else if (colm>=5&&colm<=8){ //n == 5 , n == 14-
for(byte iColm = 4,iColm2 = (byte) (colm-
4);iColm>=0;iColm--,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");}
} else {
for(byte iColm = 4,iColm2 = (byte) (colm-
4);iColm>=(colm-9);iColm--,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");}}}}
} else if ((row>=5&&row<=8)){ //n == 5 , n ==
14-
for(byte iRow = 4,iRow2 = (byte) (row-
4);iRow>=0;iRow--,iRow2++){
for(byte colm = 0;colm<14;colm++){
if(colm<5){ //n=5
for(byte iColm = colm,iColm2 =0;iColm>=0;iColm--
,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
```

```
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");
}
}else if (colm>=5&&colm<=8){ //n == 5 , n == 14-
for(byte iColm = 4,iColm2 = (byte) (colm-
4);iColm>=0;iColm--,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");
}
} else {
for(byte iColm = 4,iColm2 = (byte) (colm-
4);iColm>=(colm-9);iColm--,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");}}}}
} else {
for(byte iRow = 4,iRow2 = (byte) (row-
4);iRow>=row-9;iRow--,iRow2++){
for(byte colm = 0;colm<14;colm++){
if(colm<5){ //n=5
for(byte iColm = colm,iColm2 =0;iColm>=0;iColm--
,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");
}
}else if (colm>=5&&colm<=8){ //n == 5 , n == 14-
for(byte iColm = 4,iColm2 = (byte) (colm-
4);iColm>=0;iColm--,iColm2++){
```

```
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");
}
} else {
for(byte iColm = 4,iColm2 = (byte) (colm-
4);iColm>=(colm-9);iColm--,iColm2++){
deltaFeatureMap2[iLoop][row][colm] +=
bobotKonvolusiL3[jLoop][iLoop][iRow][iColm]*delt
aFeatureMap3[jLoop][iRow2][iColm2];
//
System.out.print("\t["+iRow+"]["+iColm+"]"+"\t["
+iRow2+"]["+iColm2+"]");}}}}}}}


//        System.out.println("Bismillah");
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_2
for(jLoop = 0;jLoop < 14;jLoop++){ //
tinggi_feture Map_2 = 42 /
tinggi_kernel_pooling_1 = 42 / 3 = 14
for(kLoop = 0;kLoop < 14;kLoop++){ //
lebar_feture Map_2 = 42 / lebar_kernel_pooling_1
= 42 / 3 = 14
deltaFeatureMap2[iLoop][jLoop][kLoop] =
deltaFeatureMap2[iLoop][jLoop][kLoop]*(1-
Math.tanh(featureMap2[iLoop][jLoop][kLoop])*Math
.tanh(featureMap2[iLoop][jLoop][kLoop]));}}}

//delta di bobot L1 fMap1 ke fMap2 Pooling
for(iLoop = 0;iLoop < 20;iLoop++){ // 20
feature_Map_2
deltaBobotPoolingL2[iLoop] = 0;
deltaBiasPoolingL2[iLoop] = 0;
for(jLoop = 0;jLoop < 14;jLoop++){ //
tinggi_feture Map_2 = 42 /
tinggi_kernel_pooling_1 = 42 / 3 = 14
```

```
for(kLoop = 0;kLoop < 14;kLoop++){ //
lebar_feture Map_2 = 42 / lebar_kernel_pooling_1
= 42 / 3 = 14
//\\Pooling//\\
for(iKern = (byte) (jLoop*3);iKern <
((jLoop+1)*3);iKern++){ //
tinggi_kernel_pooling_1 = 3
for(jKern = (byte) (kLoop*3);jKern <
((kLoop+1)*3);jKern++){ //
lebar_kernel_pooling_1 = 3
//
deltaBobotPoolingL2[iLoop] +=
deltaFeatureMap2[iLoop][jLoop][kLoop]*featureMap
1[iLoop][iKern][jKern];}}

deltaBiasPoolingL2[iLoop] +=
deltaFeatureMap2[iLoop][jLoop][kLoop];
//\\Pooling//\\}}

deltaBobotPoolingL2[iLoop] *= alpha;
deltaBiasPoolingL2[iLoop] *= alpha;}
```

5.6.5 Implementasi *Convolutional Layer* C3

1. Propagasi Maju

```
//Layer 3 Konvolusi-2
for(iLoop = 0;iLoop < 100;iLoop++){ // 100
feature_Map_3
for(jLoop = 0;jLoop < 10;jLoop++){ //
tinggi_feature_Map_3 = 46 - (5-1) = 42
for(kLoop = 0;kLoop < 10;kLoop++){ //
lebar_feature_Map_3 = 46 - (5-1) = 42
//\\Konvolusi 2//\\
featureMap3[iLoop][jLoop][kLoop] = 0;
for(mLoop = 0;mLoop < 20;mLoop++){
for(iKern = jLoop;iKern < jLoop+5;iKern++){ //
tinggi_kernel_konvolusi_2 = 5
for(jKern = kLoop;jKern < kLoop+5;jKern++){ //
tinggi_kernel_konvolusi_2 = 5
```

```
featureMap3[iLoop][jLoop][kLoop] +=
featureMap2[mLoop][iKern][jKern]*bobotKonvolusiL
3[iLoop][mLoop][iKern-jLoop][jKern-kLoop];}}}

//\\Konvolusi 2//\\
featureMap3[iLoop][jLoop][kLoop] =
Math.tanh(featureMap3[iLoop][jLoop][kLoop] +
biasKonvolusiL3[iLoop][jLoop][kLoop]);}}}
```

2. Propagasi Mundur

```
//delta di unit featureMap3(X) (bias tidak
diikutsertakan dalam perhitungan, karena bias =
1)
for(iLoop = 0;iLoop < 100;iLoop++){ // 100
feature_Map_3
for(jLoop = 0;jLoop < 10;jLoop++){ //
tinggi_feature_Map_3 = 46 - (5-1) = 42
for(kLoop = 0;kLoop < 10;kLoop++){ //
lebar_feature_Map_3 = 46 - (5-1) = 42
deltaFeatureMap3[iLoop][jLoop][kLoop] =
bobotPoolingL4[iLoop] *
deltaFeatureMap4[iLoop][jLoop/2][kLoop/2]*(1-
Math.tanh(featureMap3[iLoop][jLoop][kLoop])*Math
.tanh(featureMap3[iLoop][jLoop][kLoop]));
//\\Fungsi aktivasi anti-symmetric sigmoid
tanh//\\}}}

//delta di bobot L2 fMap2 ke fMap3 Konvolusi...
for(iLoop = 0;iLoop < 100;iLoop++){
for(mLoop = 0;mLoop < 20;mLoop++){
for(jLoop = 0;jLoop < 5;jLoop++){
for(kLoop = 0;kLoop < 5;kLoop++){
deltaBobotKonvolusiL3[iLoop][mLoop][jLoop][kLoop
] = 0;
for(iKern = 0;iKern < 10;iKern++){ //
tinggi_kernel_konvolusi_2 = 5
for(jKern = 0;jKern < 10;jKern++){ //
tinggi_kernel_konvolusi_2 = 5
```

```
deltaBobotKonvolusiL3[iLoop][mLoop][jLoop][kLoop
] +=
deltaFeatureMap3[iLoop][iKern][jKern]*featureMap
2[mLoop][iKern+jLoop][jKern+kLoop];}}

deltaBobotKonvolusiL3[iLoop][mLoop][jLoop][kLoop
] *= alpha;}}}}

for(iLoop = 0;iLoop < 100;iLoop++){
for(iKern = 0;iKern < 10;iKern++){ //
tinggi_kernel_konvolusi_2 = 5
for(jKern = 0;jKern < 10;jKern++){ //
tinggi_kernel_konvolusi_2 = 5
deltaBiasKonvolusiL3[iLoop][iKern][jKern] =
alpha*deltaFeatureMap3[iLoop][iKern][jKern];}}}
```

5.6.6 Implementasi *Pooling Layer* P4

1. Propagasi Maju

```
//Layer 4 MaxPooling-2
for(iLoop = 0;iLoop < 100;iLoop++){ // 100
feature_Map_4
for(jLoop = 0;jLoop < 5;jLoop++){ //
tinggi_feature_Map_4 = 10 /
tinggi_kernel_pooling_2 = 10 / 2 = 5
for(kLoop = 0;kLoop < 5;kLoop++){ //
lebar_feature_Map_4 = 10 /
lebar_kernel_pooling_2 = 10 / 2 = 5
//\\Pooling//\\
featureMap4[iLoop][jLoop][kLoop] = 0;
for(iKern = (byte) (jLoop*2);iKern <
((jLoop+1)*2);iKern++){ //
tinggi_kernel_pooling_2 = 2
for(jKern = (byte) (kLoop*2);jKern <
((kLoop+1)*2);jKern++){ //
tinggi_kernel_pooling_2 = 2
featureMap4[iLoop][jLoop][kLoop] +=
featureMap3[iLoop][iKern][jKern];}}
```

```
//\\Pooling//\\
featureMap4[iLoop][jLoop][kLoop] =
Math.tanh(bobotPoolingL4[iLoop]*featureMap4[iLoo
p][jLoop][kLoop]+biasPoolingL4[iLoop]);}}}
```

2. Propagasi Mundur

```
//100x5x5 = 200 x 200x100x5x5
for(iLoop = 0;iLoop < 100;iLoop++){
for(jLoop = 0;jLoop < 5;jLoop++){
for(kLoop = 0;kLoop < 5;kLoop++){
deltaFeatureMap4[iLoop][jLoop][kLoop] = 0;
for(lLoop = 0;(lLoop & 0xFF)<200;lLoop++){
deltaFeatureMap4[iLoop][jLoop][kLoop] +=
deltaZ[(lLoop & 0xFF)]*bobotXtoZ[(lLoop &
0xFF)][iLoop][jLoop][kLoop];
}
deltaFeatureMap4[iLoop][jLoop][kLoop] =
deltaFeatureMap4[iLoop][jLoop][kLoop]*(1-
Math.tanh(featureMap4[iLoop][jLoop][kLoop])*Math
.tanh(featureMap4[iLoop][jLoop][kLoop]));}}}

//delta di bobotPooling2 L3 fMap3 ke fMap4
for(iLoop = 0;iLoop < 100;iLoop++){ // 100
feature_Map_4
deltaBobotPoolingL4[iLoop] = 0;
deltaBiasPoolingL4[iLoop] = 0;
for(jLoop = 0;jLoop < 5;jLoop++){ //
tinggi_feature_Map_4 = 10 /
tinggi_kernel_pooling_2 = 10 / 2 = 5
for(kLoop = 0;kLoop < 5;kLoop++){ //
lebar_feature_Map_4 = 10 /
lebar_kernel_pooling_2 = 10 / 2 = 5
//\\Pooling//\\
for(iKern = (byte) (jLoop*2);iKern <
((jLoop+1)*2);iKern++){ //
tinggi_kernel_pooling_2 = 2
```

```
for(jKern = (byte) (kLoop*2);jKern <
((kLoop+1)*2);jKern++){ //
tinggi_kernel_pooling_2 = 2
deltaBobotPoolingL4[iLoop] +=
deltaFeatureMap4[iLoop][jLoop][kLoop]*featureMap
3[iLoop][iKern][jKern];}}

//\\Pooling//\\
deltaBiasPoolingL4[iLoop] +=
deltaFeatureMap4[iLoop][jLoop][kLoop];//*1}}

deltaBobotPoolingL4[iLoop] *= alpha;
deltaBiasPoolingL4[iLoop] *= alpha ;
}
```

5.6.7 Implementasi *Hidden Layer* H5

1. Propagasi Maju

```
for(iLoop = 0;(iLoop & 0xFF)<200;iLoop++){
z_net[(iLoop & 0xFF)] = 0;
for(jLoop = 0;jLoop<100;jLoop++){
for(kLoop = 0;kLoop<5;kLoop++){
for(lLoop = 0;lLoop<5;lLoop++){
z_net[(iLoop & 0xFF)] += bobotXtoZ[(iLoop &
0xFF)][jLoop][kLoop][lLoop]
*featureMap4[jLoop][kLoop][lLoop];}}}
z_net[(iLoop & 0xFF)] = Math.tanh(z_net[(iLoop &
0xFF)]+biasXtoZ[(iLoop & 0xFF)]);
}
```

2. Propagasi Mundur

```
//delta di unit z (bias tidak diikutsertakan
dalam perhitungan, karena bias = 1)
for(iLoop = 0;(iLoop & 0xFF)<200;iLoop++){
deltaZ[(iLoop & 0xFF)] = 0;
for(jLoop = 0;jLoop< Klas;jLoop++)
```

```
deltaZ[(iLoop & 0xFF)] +=
deltaY[jLoop]*bobotZtoY[jLoop][(iLoop & 0xFF)];
deltaZ[(iLoop & 0xFF)] = deltaZ[(iLoop &
0xFF)]*(1-Math.tanh(z_net[(iLoop &
0xFF)])*Math.tanh(z_net[(iLoop & 0xFF)]));
}

//delta di bobot x ke z
for(iLoop = 0;(iLoop & 0xFF)<200;iLoop++){

for(jLoop = 0;jLoop<100;jLoop++){
for(kLoop = 0;kLoop<5;kLoop++){
for(lLoop = 0;lLoop<5;lLoop++){
deltaBobotXtoZ[(iLoop &
0xFF)][jLoop][kLoop][lLoop] =
alpha*deltaZ[(iLoop &
0xFF)]*featureMap4[jLoop][kLoop][lLoop];}}}
deltaBiasXtoZ[(iLoop & 0xFF)] = alpha *
deltaZ[(iLoop & 0xFF)];
}
```

5.6.8 Implementasi *Output Layer* O5

1. Propagasi Maju

```
for(iLoop = 0;iLoop< Klas;iLoop++){
y_net[iLoop] = 0;
for(jLoop = 0;(jLoop & 0xFF) < 200;jLoop++){
//6_6_200_200
y_net[iLoop] += bobotZtoY[iLoop][(jLoop &
0xFF)]*z_net[(jLoop & 0xFF)];
}
//Fungsi aktivasi tanh
y_net[iLoop] = Math.tanh(y_net[iLoop] +
biasZtoY[iLoop]);//biasZtoY[iLoop][200)]

}
```

2. Propagasi Mundur

```
for(jLoop = 0;jLoop< Klas;jLoop++)
```

```
deltaY[jLoop] = (target[NLoop1][jLoop] -
y_net[jLoop])*(1-
Math.tanh(y_net[jLoop])*Math.tanh(y_net[jLoop]))
;

//delta di bobot z ke y
for(iLoop = 0;iLoop< Klas;iLoop++){
for(jLoop = 0;(jLoop & 0xFF) < 200;jLoop++){
deltaBobotZtoY[iLoop][(jLoop & 0xFF)] = alpha *
deltaY[iLoop]*z_net[(jLoop & 0xFF)];
}
deltabiasZtoY[iLoop] = alpha * deltaY[iLoop];
}
```

5.6.9 Implementasi Perhitungan MSE

```
for(iLoop = 0;iLoop< Klas;iLoop++){
error += ((target[NLoop1][iLoop]-
y_net[iLoop])*(target[NLoop1][iLoop]-
y_net[iLoop]));
}

error = error/Klas;
```

BAB VI
UJI COBA DAN EVALUASI SISTEM

Pada bab ini dijelaskan tahap-tahap uji coba berdasarkan implementasi sistem yang telah dibuat. Hasil uji coba akan dianalisis melalui proses verifikasi dan validasi sehingga dapat melakukan evaluasi pada sistem.

6.1 Dataset Uji Coba *Training*

Dataset uji coba *training* dikumpulkan dari relawan berupa data gambar wajah dengan berbagai kondisi pencahayaan dan posisi wajah masing-masing individu. Dataset dibagi menjadi 6 bagian berdasarkan tujuan pengujian. Rincian keterangan daftar *dataset* dapat dilihat pada **Tabel 6.1 - Tabel 6.6**.

Tabel 6.1. Keterangan uji coba dataset 1

Uji Coba Dataset 1	
Jumlah	3 x 10 gambar wajah
Relawan	1. Asmi Math 2012 2. Ahlan Math 2012 3. Zufar Math 2012
Ukuran	48 x 48 piksel
Type	Grayscale
Pengambilan gambar	Pencahayaan terang

Tabel 6.2. Keterangan uji coba dataset 2

Uji Coba Dataset 2	
Jumlah	3 x 10 gambar wajah
Relawan	1. Linda Math 2011 2. Izza Math 2012 3. Indah Math 2012
Ukuran	48 x 48 piksel
Type	Grayscale
Pengambilan gambar	Kurang pencahayaan

Tabel 6.3. Keterangan uji coba dataset 3

Uji Coba Dataset 3	
Jumlah	5 x 36 gambar wajah
Relawan	1. Suef Math 2012 2. Yusuf Math 2012 3. Resi Math 2012 4. Hendra Math 2012 5. Zufar Math 2012
Ukuran	48 x 48 piksel
Type	Grayscale
Pengambilan gambar	Kurang pencahayaan

Tabel 6.4. Keterangan uji coba dataset 4

Uji Coba Dataset 4	
Jumlah	70 gambar wajah relawan dan 70 gambar wajah bebas
Relawan	Zufar Math 2012
Ukuran	48 x 48 piksel
Type	Grayscale
Pengambilan gambar	Pencahayaan terang

Tabel 6.5. Keterangan uji coba dataset 5

Uji Coba Dataset 5	
Jumlah	6x62 gambar wajah
Relawan	1. Ibunda 2. Nenek 3. Najwa 4. Zufar
Ukuran	48 x 48 piksel
Type	Grayscale
Pengambilan gambar	Pencahayaan terang

Tabel 6.6. Keterangan uji coba dataset 6

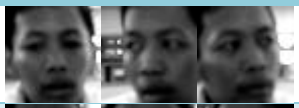
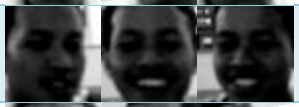
Uji Coba Dataset 6	
Jumlah	6x62 gambar wajah
Relawan	1. Ardian 2. Ibunda 3. Nenek 4. Zufar
Ukuran	48 x 48 piksel
Type	Grayscale
Pengambilan gambar	Pencahayaan terang

6.2 Uji Coba Data *Preprocessing*

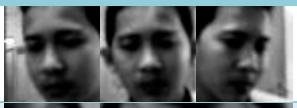
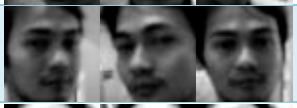
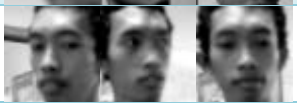
6.4.1 Ekstraksi *Extended Local Binary Pattern*

Dari dataset training wajah orang dari beberapa sudut ditranformasikan dengan metode *extended local binary pattern*. Hasil transformasi pada gambar menunjukkan bahwa kontur wajah antara lain seperti garis, tepi,dan kontur lainnya terlihat jelas dan hasil ekstraksi dapat mengatasi masalah penagaruh cahaya. Beberapa hasil ekstraksi ELBP dapat dilihat pada **Tabel 6.7 – Tabel 6.9.**

Tabel 6.8. Hasil ekstraksi ELBP dataset 1

Dataset 1		
Nama	**Gambar**	**ELBP**
Ahlan		
Asmi		
Zufar		

Tabel 6.9. Hasil ekstraksi ELBP dataset 3

Dataset 3		
Nama	**Gambar**	**ELBP**
Suef		
Yusuf		
Resi		
Hendra		
Zufar		

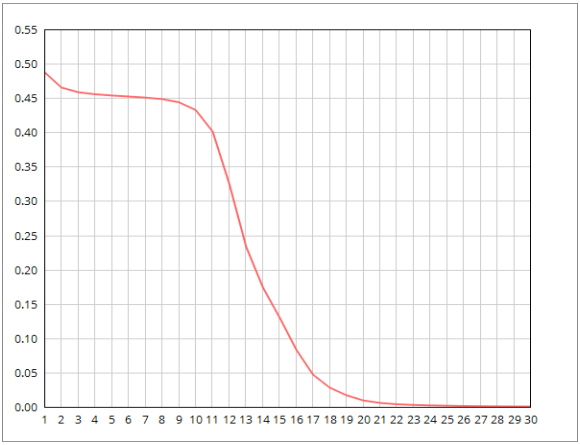
Tabel 6.10. Hasil ekstraksi ELBP dataset 4

Dataset 4			
Jenis	**Nama**	**Gambar**	**ELBP**
Positif	Zufar		
Negatif	Prisma		
	Fata		
	Harits		

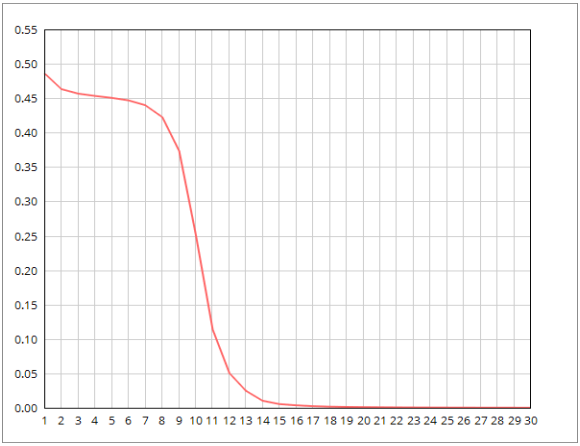
6.3 Uji Coba *Training* Model

Dataset yang telah dikumpulkan akan dilakukan uji coba *training* untuk mendapatkan model pengenalan wajah dengan benar sesuai *dataset* yang dimasukkan. Pada tahap *training* model, diperlukan input berupa *dataset* dan nama label. *Dataset* akan otomatis bersesuaian dengan nama label. Nama label yang bersesuain direpresentasikan sebagai deretan target bipolar. Hasil akhir *training* Dataset 1 menghasilkan model 1, dataset 2 menghasilkan model 2, dataset 3 menghasilkan model 3, dataset 4 menghasilkan model 4, dataset 5 menghasilkan model 5, dan dataset 6 menghasilkan model 6. Setiap keluaran model memiliki nama label dan parameter(bobot dan bias) yang sudah dilatih.

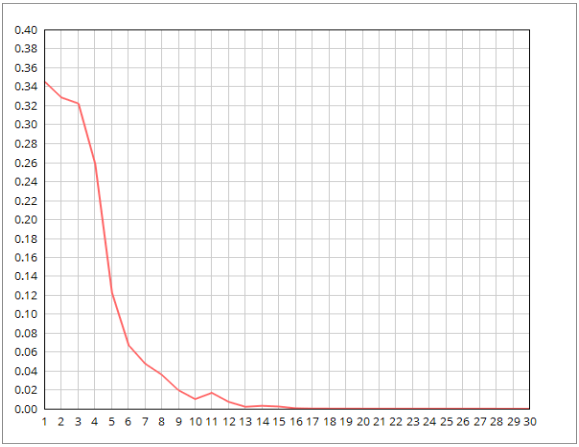
Kurva perubahan nilai *error* pada uji coba *training* dataset 1, 2, dan 3 dengan memberikan nilai alpha = 0.01 dan *error minimal = 0.05* dapat dilihat pada **Gambar 6.1**, **Gambar 6.2**, dan **Gambar 6.3**.



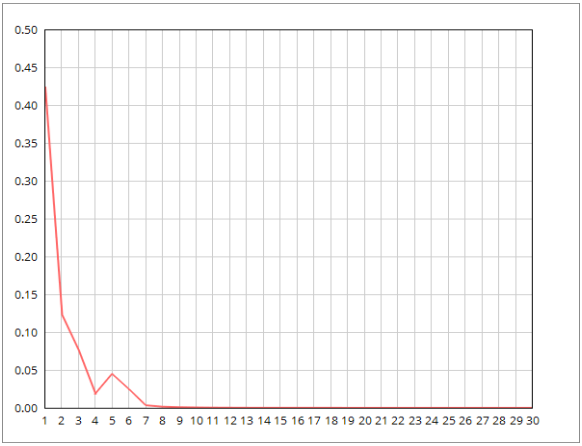
Gambar 6.1. Kurva *konvergensi training* dataset 1



Gambar 6.2. Kurva *konvergensi training* dataset 2

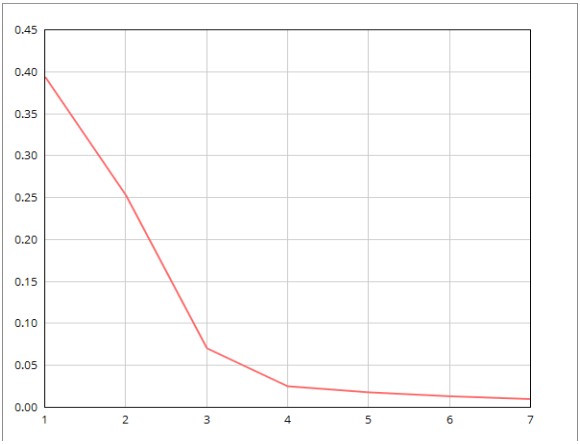


Gambar 6.3. Kurva *konvergensi training* dataset 3

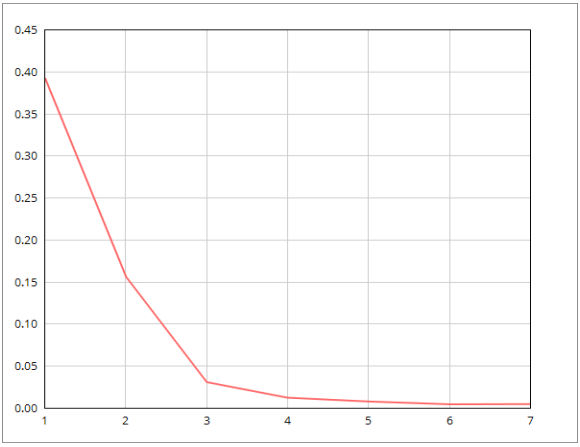


Gambar 6.4. Kurva *konvergensi training* dataset 4

Berdasarkan kurva *training* dataset 1,2, 3, dan 4 diatas bahwa jaringan sudah berhasil konvergen ke suatu titik dan seiring berjalannnya iterasi perubahan nilai *error* semakin lama semakin kecil sehingga nilai *error* mendekati konstan yaitu pada *error minimal* (konvergen dan stabil). Dataset 1 membutuhkan 29 kali iterasi, dataset 2 membutuhkan 27 kali iterasi, dataset 3 membutuhkan 25 iterasi, dam dataset 4 membuthkan 13 iterasi untuk mencapai *error minimal* sehingga stabil. Sedangkan dataset 5 dan 6 digunakan untuk percobaan *training* dengan 7 iterasi dan inisialisasi alpha =0.01 dapat dilihat pada **Gambar 6.4** dan **Gambar 6.5.**



Gambar 6.5. Kurva *konvergensi training* dataset 5

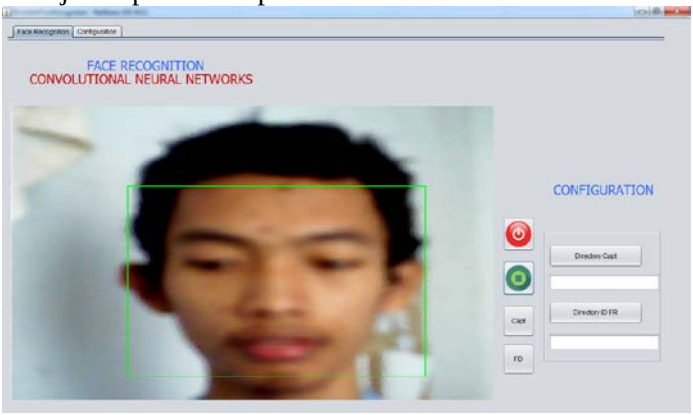


Gambar 6.6. Kurva *konvergensi training* dataset 6

Berdasarkan gambar kurva *training* dataset 5 dan 6 dalam hanya 7 kali iterasi diatas bahwa jaringan sudah berhasil konvergen ke suatu titik dan seiring berjalannnya iterasi perubahan nilai *error* semakin lama semakin kecil sehingga nilai *error* mendekati konstan yaitu pada *error minimal* (konvergen dan stabil).

6.4 Uji Coba Model

Uji coba model dilakukan dengan menjalankan implementasi pengenalan wajah secara *real-time*. Pertama aplikasi dijalankan, kemudian kamera dijalankan secara *real-time* beserta deteksi multi wajah dapat dilihat pada **Gambar 6.7.**



Gambar 6.7. Deteksi Wajah secara *real-time*

Direktori file model dicari dan dimasukkan sebagai input. Proses pengenalan dijalankan dengen menangkap gambar wajah yang terdeteksi, dirubah kedalam *grayscale* 48x48. Gambar *grayscale* akan dilakukan preprocessing ELBP menjadi ekstraksi fitur gambar. Hasil ekstraksi akan masuk model dan dilakukan klasifikasi data Keluaran kalsifikasi model berupa nama label yang akan ditampilkan pada layar monitor tepatnya di atas kiri lokasi *bounding box* terdeteksi wajah.

6.4.1 Uji Coba Model 1

Uji coba model dilakukan dengan 3 orang relawan perempuan sesuai relawan *dataset* asal model. Deskripsi pengujian model 1 ini dapat dilihat pada **Tabel 6.11**.

Tabel 6.11. Deskripsi Pengujian Model 1

Uji Model 1	
Relawan	1. Asmi Math 2012 2. Ahlan Math 2012 3. Zufar Math 2012
Kondisi lingkungan perekaman	Pencahayaan terang
Tujuan	Pengujian dan analisis tingkat akurasi dari dataset wajah laki-laki dengan kondisi pencahayaan terang

Perhitungan tingkat akurasi model 1 disajikan pada **Tabel 6.12.** Dari hasil perhitungan didapatkan hasil wajah terdeteksi sebanyak 96 dan dikenali dengan benar sebanyak 88. Perhitungan akurasi dilakukan dengan membagi jumlah terdeksi dengan jumlah dikenali dengan benar. Didapatkan tingkat akurasi sebesar 91.67 %.

Tabel 6.12. Perhitungan Tingkat Akurasi Model 1

No	Nama	Benar	Salah	Total Terdeteksi
1	Ahlan	45	1	46
2	Asmi	23	4	27
3	Zufar	20	3	23
		88	8	96
Akurasi		91.67%	8.33%	100%

Kesalahan pengenalan cenderung dikarenakan model mendapatkan gambar blur ketika kamera mengambil hasil deteksi gambar wajah bersamaan dengan relawan bergerak. Pada saat relawan berhenti bergerak, kamera menangkap gambar lebih jelas sehingga wajah dikenali dengan benar.

6.4.2 Uji Coba Model 2

Uji coba model dilakukan dengan 3 orang relawan perempuan sesuai relawan *dataset* asal model. Deskripsi pengujian model 2 ini dapat dilihat pada **Tabel 6.13.**

Tabel 6.13. Deskripsi Pengujian Model 2

Uji Model 2	
Relawan	1. Linda Math 2012 2. Izza Math 2012 3. Indah Math 2012
Kondisi lingkungan perekaman	Kurang pencahayaan
Tujuan	Pengujian dan analisis tingkat akurasi dari dataset wajah perempuan dengan kondisi kurang pencahayaan

Perhitungan tingkat akurasi model 2 disajikan pada **Tabel 6.14.** Dari hasil perhitungan didapatkan hasil wajah terdeteksi sebanyak 130 dan dikenali dengan benar sebanyak 105. Perhitungan akurasi dilakukan dengan membagi jumlah terdeksi dengan jumlah dikenali dengan benar. Didapatkan tingkat akurasi sebesar 80.15%.

Tabel 6.14. Perhitungan Tingkat Akurasi Model 2

No	Nama	Benar	Salah	Total Terdeteksi
1	Linda	40	0	40
2	Izza	37	10	47
3	Indah	28	16	44
		105	26	130
Akurasi		80.15%	19.85%	100%

6.4.3 Uji Coba Model 3

Uji coba model dilakukan dengan 5 orang relawan sesuai relawan *dataset* asal model. Deskripsi pengujian model 3 ini dapat dilihat pada **Tabel 6.15.**

Tabel 6.15. Deskripsi Pengujian Model 3

Uji Coba Model 3	
Relawan	1. Suef Math 2012 2. Yusuf Math 2012 3. Resi Math 2012 4. Hendra Math 2012 5. Zufar Math 2012
Kondisi dalam pengambilan gambar	Kurang pencahayaan
Tujuan	Pengujian dan analisis tingkat akurasi dengan kondisi kurang pencahayaan

Perhitungan tingkat akurasi model 3 disajikan pada **Tabel 6.16.** Dari hasil perhitungan didapatkan hasil wajah terdeteksi sebanyak 170 dan dikenali dengan benar sebanyak 151. Perhitungan akurasi dilakukan dengan membagi jumlah terdeksi dengan jumlah dikenali dengan benar. Didapatkan tingkat akurasi sebesar 88.62 %.

Tabel 6.16. Perhitungan Tingkat Akurasi Model 3

No	Nama	Benar	Salah	Total Terdeteksi
1	Suef	18	4	22
2	Yusuf	26	3	29
3	Resi	28	7	35
4	Hendra	19	5	24
5	Zufar	60	0	60
		151	19	170
Akurasi		88.82%	11.18%	100%

Kesalahan pengenalan cenderung dikarenakan perekaman *real-time* pada kondisi kurang lingkungan pencahayaan sehingga didapatkan gambar kurang tajam.

6.4.4 Uji Coba Model 4

Uji coba model dilakukan dengan 2 orang relawan. Deskripsi pengujian model 4 ini dapat dilihat pada **Tabel 6.17.**

Tabel 6.17. Deskripsi Pengujian Model 4

Uji Coba Model 4	
Relawan	Zufar Math 2012 Asna math 2013
Kondisi dalam pengambilan gambar	Pencahayaan terang
Tujuan	Pengujian dan analisis tingkat akurasi pada obyek wajah yang tidak dikenali

Perhitungan tingkat akurasi model 4 disajikan pada **Tabel 6.18.** Dari hasil perhitungan didapatkan hasil w ajah terdeteksi sebanyak 159 dan dikenali dengan benar sebanyak 142. Perhitungan akurasi dilakukan dengan membagai jumlah terdeksi dengan jumlah dikenali dengan benar. Didapatkan tingkat akurasi sebesar 89.30%.

Tabel 6.18. Perhitungan Tingkat Akurasi Model 4

No	Nama	Benar	Salah	Total Terdeteksi
1	Zufar	77	14	91
2	Tidak Dikenali	65	3	68
		142	17	159
Akurasi		89.30%	10.70%	100%

Model mampu mengenali obyek yang tidak dikenali. Hal ini dapat dilakukan jika dimasukkan dataset sembarang wajah kecuali wajah yang sudah masuk kelas lain.

6.4.5 Uji Coba Model 5

Uji coba model dilakukan dengan 4 orang relawan. Deskripsi pengujian model 5 ini dapat dilihat pada **Tabel 6.19.**

Tabel 6.19. Deskripsi Pengujian Model 5

Uji Coba Model 5	
Relawan	1. Ibunda 2. Nenek 3. Najwa 4. Zufar
Kondisi dalam pengambilan gambar	Kurang pencahayaan
Tujuan	Pengujian dan analisis tingkat akurasi dengan kondisi kurang pencahayaan

Perhitungan tingkat akurasi model 5 disajikan pada **Tabel 6.20.** Dari hasil perhitungan didapatkan hasil w ajah terdeteksi sebanyak 837 dan dikenali dengan benar sebanyak 660 . Perhitungan akurasi dilakukan dengan membagi jumlah terdeksi dengan jumlah dikenali dengan benar. Didapatkan tingkat akurasi sebesar 78.86%.

Tabel 6.20. Perhitungan Tingkat Akurasi Model 5

No	Nama	Benar	Salah	Total Terdeteksi
1	Ibunda	138	23	161
2	Nenek	170	12	182
3	Najwa	195	94	289
4	Zufar	157	48	205
		660	177	837
Akurasi		78.86%	21.14%	100%

Akurasi model 5 paling kecil dibandingakan dengan model lainnya. Hal ini disebabkan karena pada proses *training* mendapatkan model hanya dilakukan 7 iterasi saja walaupun dataset lebih besar dibandingkan data lainnya.

6.4.6 Uji Coba Model 6

Uji coba model dilakukan dengan 4 orang relawan. Deskripsi pengujian model 6 ini dapat dilihat pada **Tabel 6.21.**

Tabel 6.21. Deskripsi Pengujian Model 6

Uji Coba Model 6	
Relawan	1. Ardian 2. Ibunda 3. Nenek 4. Zufar
Kondisi dalam pengambilan gambar	Pencahayaan terang
Tujuan	Pengujian dan analisis tingkat akurasi dengan kondisi pencahayaan terang

Perhitungan tingkat akurasi model 5 disajikan pada **Tabel 6.22.** Dari hasil perhitungan didapatkan hasil wajah terdeteksi sebanyak 170 dan dikenali dengan benar sebanyak 140. Perhitungan akurasi dilakukan dengan membagai jumlah terdeksi dengan jumlah dikenali dengan benar. Didapatkan tingkat akurasi sebesar 82.35%.

Tabel 6.22. Perhitungan Tingkat Akurasi Model 6

No	Nama	Benar	Salah	Total Terdeteksi
1	Ardian	41	22	63
2	Ibunda	43	5	48
3	Nenek	29	1	30
4	Zufar	27	2	29
		140	30	170
Akurasi		82.35%	17.65%	100%

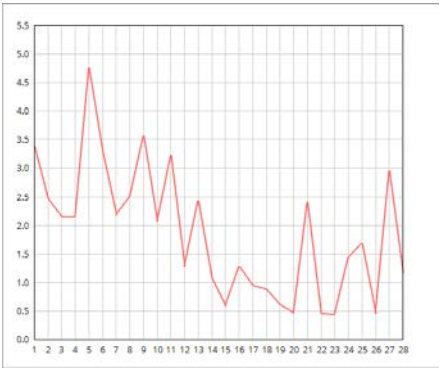
Akurasi model tidak begitu tinggi, hal ini sama juga disebabkan karena pada proses *training* untuk mendapatkan model 6 hanya dilakukan 7 iterasi saja.

Hasil keseluruhan ujicoba didapatkan bahwa model dapat melakukan klasifikasi dengan benar, akan tetapi beberapa wajah salah diklasifikasi. Secara keseluruhan akurasi seluruh model didapatkan rata-rata akurasi:

$$\text{Akurasi} = \frac{m\,1 + m\,2 + m\,3 + m\,4}{6}$$

$$= \frac{91.67 + 80.15 + 88.82 + 89.30}{6}\%$$
$$= 87.48\%$$

Selanjutnya akan di uji coba perbandingan antara inisialisasi dengan menggunakan standar distribusi normal dengan tanpa standar distribusi normal. Dataset 2 dipilih sebagai pembanding dalam uji coba ini. Pada uji coba sebelumnya dataset 2 sudah digunakan inisialisasi dengan standar distribusi normal, kemudian uji coba hanya dilakukan dengan inisialisasi dengan tanpa standar distribusi normal atau acak yang disajikan pada **Gambar 6.8.**



Gambar 6.8. Kurva *konvergensi training* dataset 2 inisialisasi dengan tanpa standar distribusi normal

Berdasarkan gambar kurva *training* dataset 2 dengan inisialisasi tanpa standar distribusi normal diatas bahwa jaringan tidak dapat stabil. Walaupun beberapa saat sampai iterasi ke 20 konvergen akan tetapi pada iterasi selanjutnya *error* naik lagi yang menyebabkan secara keseluruhan *training* tidak konvergen. Sehingga jaringan lebih baik menggunakan inisialisasi dengan standar distribusi normal untuk mencapai konvergen dan stabil dalam proses *training.*

Dari seluruh hasil uji coba didapatkan beberapa sebagai berikut:

1. Tingkat akurasi lebih tergantung pada jumlah iterasi *training* dari pada jumlah dataset. Hal ini dibuktikan dari ujicoba

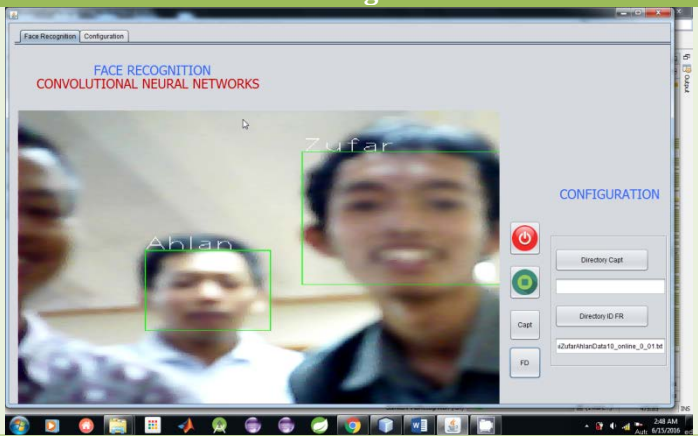
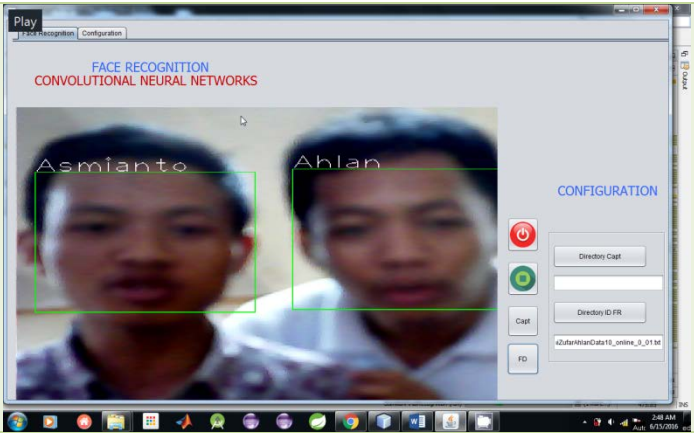
model 1, 2, 3, dan 4 dari jumlah dataset sedikit memiliki akurasi yang lebih tinggi dibandingkan dataset 5 dan 6 karena lebih banyak dilakukan iterasi.

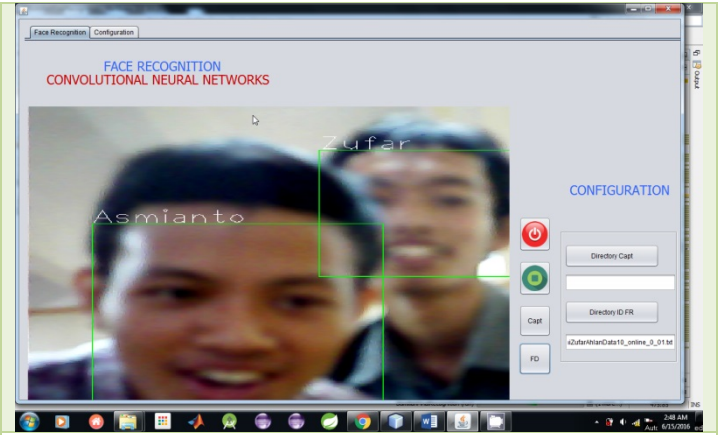
2. Intesitas cahaya baik pada dataset maupun pada saat perekaman sangat berpengaruh pada akurasi. Intesitas cahaya terang mendapatkan akurasi yang lebih tinggi dibandingkan gambar yang memiliki intesitas cahaya kecil. Hal ini dapat dilihat dari perbandingan model 1 dengan model 3 dan model 6 dengan model 5.
3. Kemampuan kamera dalam menangkap gambar bergerak sangat menentukan tingkat akurasi. Hasil penangkapan gambar dengan kamera yang memiliki fokus tinggi menghasilkan gambar yang leibh jelas sehingga data yang diperoleh lebih jelas dari pada kamera yang kurang memiliki fokus dalam pengambilan gambar.

Inisialisasi dengan standar distribusi normal lebih baik digunakan untuk mendapatkan kondisi stabil dan konvergen dari pada tanpa standar distribusi normal yang dapat mengakibatkan *training* berjalan lama tanpa adanya konvergensi *error* yang jelas.

LAMPIRAN A
Hasil Uji Coba Pengenalan

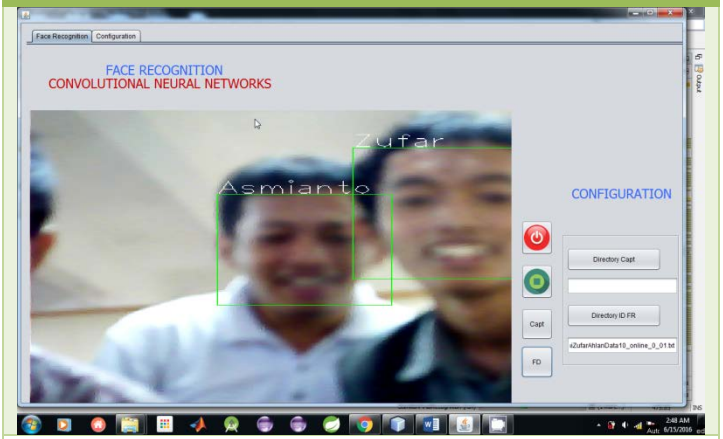
Tabel A.1. Hasil pengenalan wajah secara *real-time* menggunakan model 1

Dikenali dengan Benar

Ahlan(kiri) dan Zufar(kanan)

Asmianto (kiri) dan Ahlan (kanan)



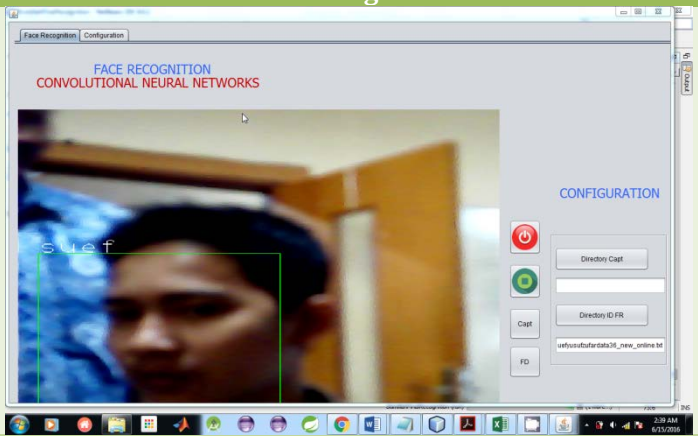
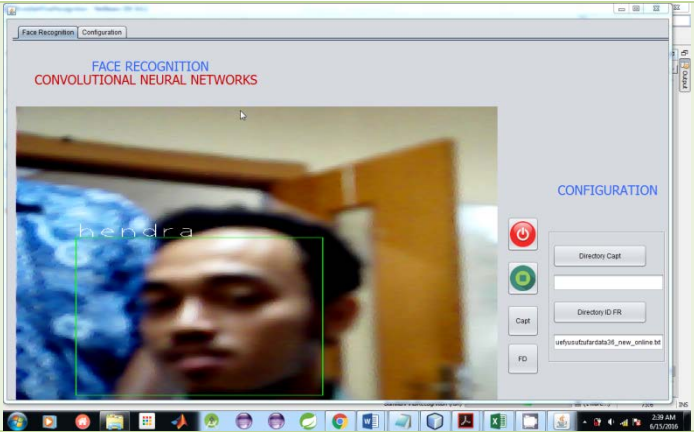
Asmianto(kiri) dan Zufar(kanan)

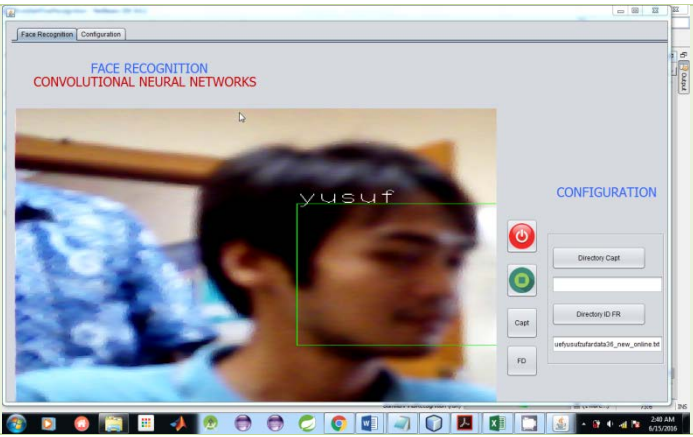
Salah Dikenali



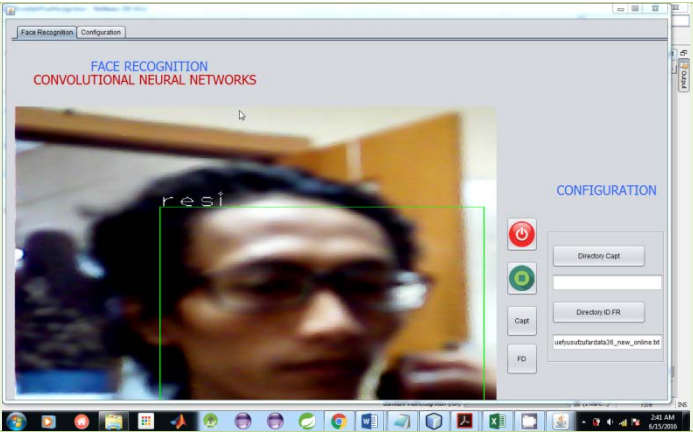
Ahlan(kiri) dan Zufar(kanan)

Tabel A.2. Hasil pengenalan wajah secara *real-time* menggunakan model 3

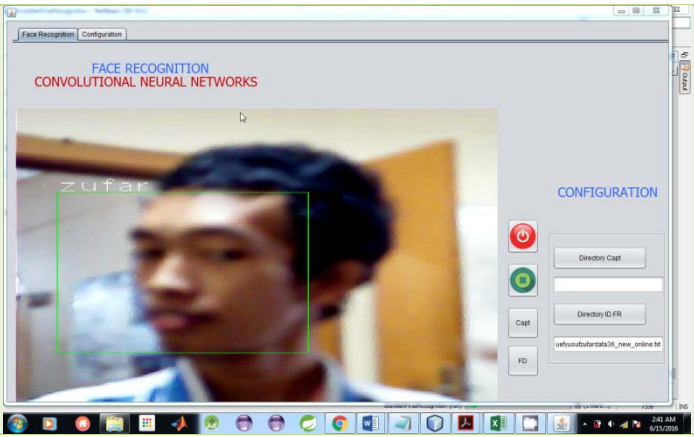
Dikenali dengan Benar

Muhammad Suef

Hendra Martono



Yusuf



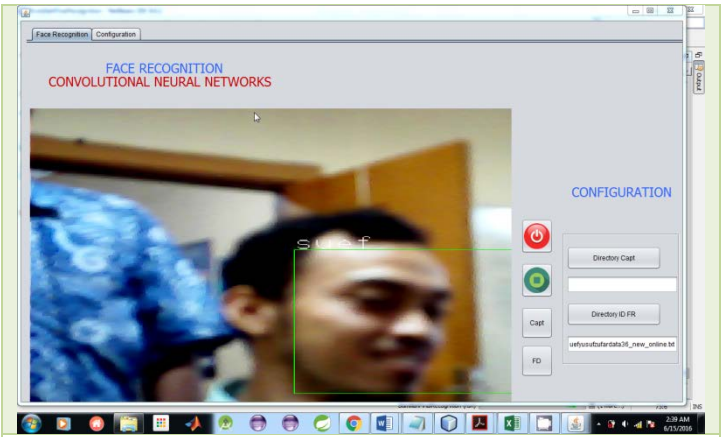
Resi Arumin Sani



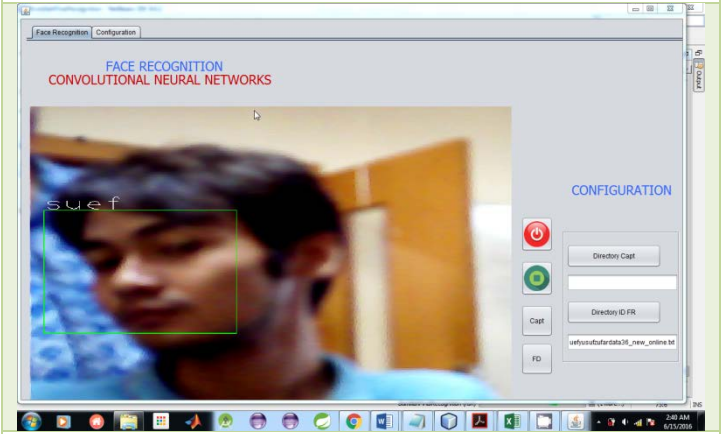
Muhammad Zufar

Salah Dikenali

Muhamad Suef



Hendra Martono

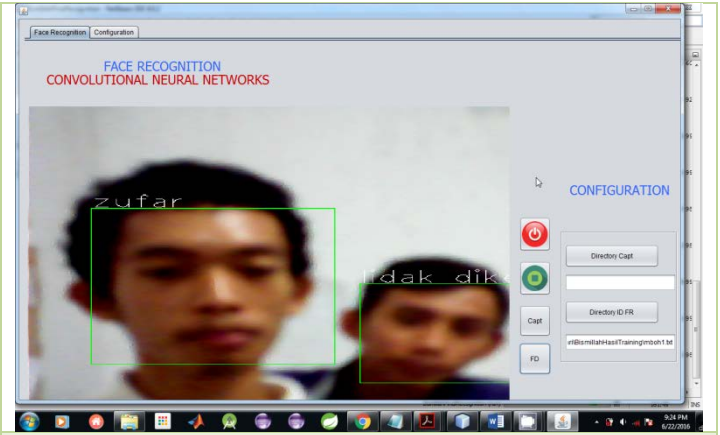


Yusuf

Resi Arumin Sani

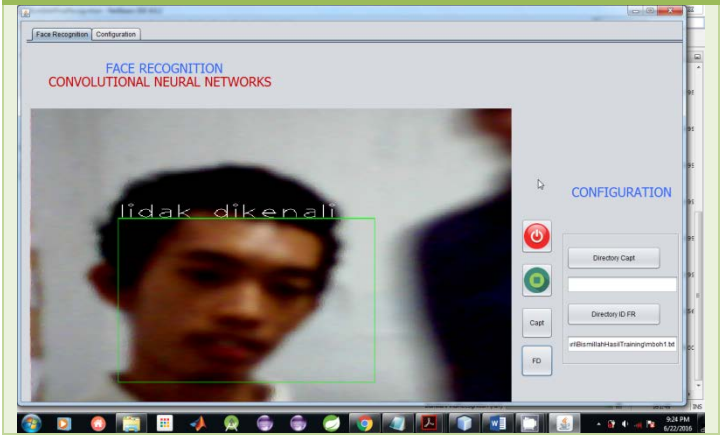
Tabel A.3. Hasil pengenalan wajah secara *real-time* menggunakan model 4

Dikenali dengan Benar

Zufar dan Asna(Obyek tidak diketahui)

Salah Dikenali



BAB VII
KESIMPULAN DAN SARAN

7.1 Kesimpulan

Dari uji coba dan evaluasi pada bab sebelumnya, dapat disimpulkan beberapa hal sebagai berikut:

1. Konstruksi model *Convolutional Neural Networks* dengan kedalaman 7 *layer* model konvolusi sebagai pembangun jaringan antara lain *input layer*, *convolutional layer C1*, *pooling layer P2*, *convolutional layer C3*, *pooling layer P4*, *hidden layer H* dan *output layer F* berhasil mengklasifikasikan gambar wajah dengan rata-rata tingkat akurasi lebih dari 87%.
2. Penggunaan ekstraksi *Extended Local Binary Pattern* mampu mengatasi pengaruh intensitas cahaya pada gambar sehingga gambar yang terkena gangguan berupa intensitas cahaya dapat menghasilkan ekstraksi pola fitur yang hampir sama dengan gambar yang mendapatkan pencahayaan rendah dan konfigurasi inisialisasi parameter bobot dengan menggunakan persebaran terdistribusi normal standar dapat mempercepat konvergensi dan kestabilan dibandingkan melakukan inisialisasi secara acak.

7.2 Saran

Saran yang diberikan untuk perbaikan dan kelanjutan dari penelitian ini adalah:

1. Memilih *hardware* kamera digital yang memiliki resolusi tinggi dan memiliki fitur *autofocus* agar didapatkan gambar yang jelas walaupun obyek bergerak sehingga dapat meningkatkan kinerja jaringan sebelum masuk model baik untuk deteksi dan pengenalan.
2. Melakukan eksplorasi terhadap kedalaman jaringan *Convolutional Neural Networks*, jenis *pooling layer*, fungsi aktivasi, dan jenis metode klasifikasi.

3. Menambahkan metode untuk menghindari *overfitting* pada jaringan.
4. Menggunakan tipe pelatihan *batch training* pada jaringan.

DAFTAR PUSTAKA

[1] T. Dunstone and N. Yager. *Biometric System and Data Analysis*: Springer, 2009.

[2] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998d). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278–2324.

[3] M. Turk, A. Pentland. *Eigenfaces for recognition*. J. Cognitive Neurosci. 3 (1) (1991) 71–86.\

[4] X. Wang and X. Tang. *Dual-Space Linear Discriminant Analysis for Face Recognition*. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. II-564 - II-569.

[5] G. Guo, S. Z. Li, and K. Chan. *Face Recognition by Support Vector Machines*. In Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000, pp. 196-201.

[6] T. Ojala, M. Pietikäinen, T. Mäenpää. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 971–987.

[7] C. Fernandez, M.A. Vicente. *Face recognition using multiple interest point detectors and sift descriptors*. In: Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition. 2008, pp. 1–7.

[8] C. Garcia and M. Delakis. *Convolutional face finder: a neural architecture for fast and robust face detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, pp. 1408-1423, 2004.

[9] Fok Hing Chi and A. Bouzerdoum. *A Gender Recognition System using Shunting Inhibitory Convolutional Neural Networks*. In International Joint Conference on Neural Networks, 2006, pp. 5336-5341.

[10] F. J. Huang and Y. LeCun. *Large-scale Learning with SVM and Convolutional Nets for Generic Object Categorization*.

In Proceedings of Computer Vision and Pattern Recognition Conferenc e, 2006.

[11] Gonzalez, R.C and Rafael E.W. *Digital Image Processing.* Prentice-Hall. Inc.. United State, America. 2008.

[12] Timo Ahonen, Abdenour Hadid,and Matti Pietik¨ ainen. *Face Description with Local Binary Patterns:Application to Face Recognition.* vol. 28 no. 12, pp. 2037-2041, December 2006.

[13] Mac Developer Library. *Performing Convolution Operations[Online]*. Available: https://developer.apple.com /library/mac/documentation.

[14] Hubel, D. and Wiesel, T. (1968). *Receptive fields and functional architecture of monkey striate cortex.* Journal of Physiology (London), 195, 215–243.

[15] Fukushima, K. (1980*). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.* Biological Cybernetics, 36, 193–202.

[16] Serre, T., Wolf, L., Bileschi, S., and Riesenhuber, M. (2007). *Robust object recog- nition with cortex-like mechanisms.* IEEE Trans. Pattern Anal. Mach. Intell., 29(3), 411–426. Member-Poggio, Tomaso.

[17] Han, Jun; Morag, Claudio (1995). *The influence of the sigmoid function parameters on the speed of backpropagation learning.* In Mira, José; Sandoval, Francisco. From Natural to Artificial Neural Computation.

[18] Y. LeCun, I. Kanter, and S.A.Solla. *Second-order properties of error surfaces: learning time and generalization.* Advances in Neural Information Processing Systems, vol. 3, pp. 918-924, 1991.

[19] Vinod Nair and Geoffrey Hinton (2010). *Rectified linear units improve restricted Boltzmann machines.*

[20] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2014). *Rectifier Nonlinearities Improve Neural Network Acoustic Models*.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.Delving Deep into Rectifiers: *Surpassing Human-Level Performance on ImageNet Classification*. 2015.

BIODATA PENULIS

Penulis memiliki nama lengkap Muhammad Zufar, lahir di Nganjuk pada tanggal 22 Agustus 1995. Penulis berasal dari Kota Nganjuk, bertempat tinggal di Ds. Dadapan RT.01/RW.02 Kec. Ngronggot, Kab. Nganjuk. Pendidikan formal yang pernah ditempuh yaitu SD N 1 DADAPAN, SMP N 1 NGRONGGOT, dan SMA N 1 KERTOSONO. Kemudian penulis melanjutkan studi di jurusan Matematika ITS, dengan bidang minat ilmu komputer. Dalam bidang minat ini penulis mulai mengenal bahasa pemrograman diantaranya adalah C, C++, Java, dan MATLAB. Semasa menempuh jenjang pendidikan S-1, penulis juga berorganisasi di KM ITS melalui LDJ MATEMATIKA ITS yang bernama IBNU MUQLAH sebagai Ketua Dep. Syiar (2014-2015). Selain itu, penulis juga melaksanakan Kerja Praktek di PT. Multipolar di bagian pengembangan web sistem informasi pada tahun 2015. Selama penulisan Tugas Akhir ini Penulis tidak lepas dari kekurangan, untuk itu penulis mengharapkan kritik, saran, dan pertanyaan mengenai Tugas Akhir ini yang dapat dikirimkan melalui *e-mail* ke zufar.muh@gmail.com.